A STUDY

OF

THE MATHEMATICS OF DEEP LEARNING

---

by

Anirbit Mukherjee

A dissertation submitted to Johns Hopkins University in conformity with the requirements for the degree of Doctor of Philosophy.

Baltimore, Maryland

July, 2020

This thesis is dedicated to my mother Dr. Suranjana Sur ("Mame").

Long before I had started learning arithmetic in school, my mother got me a geometry box and taught me how to construct various angles using a compass. Years before I had started formally studying any of the natural sciences, she created artificial clouds inside the home and taught me about condensation and designed experiments with plants to teach me how they do respiration. This thesis began with the scientific knowledge that my mother imparted regularly to me from right since I was a kid.

Thesis Abstract

"Deep Learning"/"Deep Neural Nets" is a technological marvel that is now increasingly deployed at the cutting-edge of artificial intelligence tasks. This ongoing revolution can be said to have been ignited by the iconic 2012 paper from the University of Toronto titled "ImageNet Classification with Deep Convolutional Neural Networks" by Alex Krizhevsky, Ilya Sutskever and Geoffrey E. Hinton. This paper showed that deep nets can be used to classify images into meaningful categories with almost human-like accuracies! As of 2020 this approach continues to produce unprecedented performance for an ever widening variety of novel purposes ranging from playing chess to self-driving cars to experimental astrophysics and high-energy physics. But this new found astonishing success of deep neural nets in the last few years has been hinged on an enormous amount of heuristics and it has turned out to be extremely challenging to be mathematically rigorously explainable. In this thesis we take several steps towards building strong theoretical foundations for these new paradigms of deep-learning.

Our proofs here can be broadly grouped into three categories,

- **Understanding Neural Function Spaces** We show new circuit complexity theorems for deep neural functions over real and Boolean inputs and prove classification theorems about these function spaces which in turn lead to exact algorithms for empirical risk minimization for depth 2 ReLU nets.

  We also motivate a measure of complexity of neural functions and leverage techniques from polytope geometry to constructively establish the existence of high-complexity neural functions.

- **Understanding Deep Learning Algorithms** We give fast iterative stochastic algorithms which can learn near optimal approximations of the true parameters of a ReLU gate in the realizable setting. (There are improved versions of this result available in our papers Mukherjee and Muthukumar, 2020b; Mukherjee and Muthukumar, 2020a which are not included in the thesis.)

We also establish the first ever (a) mathematical control on the behaviour of noisy gradient descent on a ReLU gate and (b) proofs of convergence of stochastic and deterministic versions of the widely used adaptive gradient deep-learning algorithms, RMSProp and ADAM. This study also includes a first-of-its-kind detailed empirical study of the hyper-parameter values and neural net architectures when these modern algorithms have a significant advantage over classical acceleration based methods.

- **Understanding The Risk Of (Stochastic) Neural Nets** We push forward the emergent technology of PAC-Bayesian bounds for the risk of stochastic neural nets to get bounds which are not only empirically smaller than contemporary theories but also demonstrate smaller rates of growth w.r.t increase in width and depth of the net in experimental tests. These critically depend on our novel theorems proving noise resilience of nets.

  This work also includes an experimental investigation of the geometric properties of the path in weight space that is traced out by the net during the training. This leads us to uncover certain seemingly uniform and surprising geometric properties of this process which can potentially be leveraged into better bounds in future.

"Study hard what interests you the most in the most undisciplined, irreverent, and original manner possible."

- Richard Feynman

Acknowledgements

Sometime in Fall 2015, I landed at Prof. Amitabh Basu's office as a jobless, penniless and homeless grad student looking for a fresh start in life while on leave from my previous physics grad school and my immigration documents being in a complete mess. My JHU story started in the middle of me giving up on academics as a whole. Only a couple of weeks before I had that out-of-the-blue meeting with my adviser Prof. Amitabh Basu, I was essentially putting up in a room in the attic of my friend Dr. Pooja Tyagi and her husband Dr. Lars Grant in the suburbs of Urbana-Champaign - because my lease there had run out and without a job I couldn't have rented a new place.

An year before this fortuitous meeting with Prof. Amitabh Basu, I was a jobless guy sharing a room with a bunch of strangers in a seedy hotel in a dark alley of Bangalore, India. I was roaming around the city trying to meet scientists there. (Long back during my undergrad I had once landed in Bangalore at the "National Centre for Biological Sciences (NCBS)" looking to attend a conference on evolutionary biology - while I was considering doing that for research!) It was during these random travels in summer 2014 that I landed a meeting with the legendary mathematician Prof. Nikhil Srivastava (now at UC Berkeley) who was then at Microsoft Research, Bangalore. It was Prof. Nikhil Srivastava, who over lunch gave me the final push to switch paths away from physics and start doing mathematics full-time. Most critically Prof. Nikhil Srivastava got me a TA position with Prof. Alexandra Kolla in the Computer Science department of UIUC. (I do not think that ever before this the CS department of UIUC had ever hired a physics student as a TA! On many days I would read up multiple chapters of the book by Cormen, Leiserson, Rivest and Stein to be able to teach the undergrad class the next day!)

This arrangement with Prof. Alexandra Kolla, gave me a funding mechanism which I used to take up every advanced course in theoretical CS that was on offer at UIUC and via participating in the research meetings with her I got my first glimpse of what "applied mathematics" looks like! Before this I had never seen anything about algorithms and invariably I bombed at the mid-semester exam of the extremely advanced graduate algorithms course of UIUC that I had taken up. But with enormous effort I managed to pass that course and mustered up the courage to register for the graduate complexity theory course that was being taught by Prof. Mahesh Vishwanathan. That was indeed a

turning point given how much I enjoyed the subject compared to almost everything that I had studied before this. Prof. Mahesh kept encouraging me and he kept me going through thick and thin. I shall forever be indebted to Prof. Mahesh Vishwanathan for agreeing to write letters of recommendation for this complete outsider that I was and for so critically helping me restart a career outside physics!

I stumbled into deep-learning while I randomly went to NYC to attend a conference on the same while not even knowing what a neural net is. The first time I listened to the talks, particularly by Prof. Anna Choromanska (NYU), on what the theoretical puzzles here are, I was somehow very strongly reminded of the nature of questions in String Theory that I so wanted to pursue while in physics.

It was only much later that I came to know of the startling fact that arguably neural nets can be said to have majorly pioneered in Johns Hopkins University itself thanks to the series of historic papers that were co-authored by the living legend Terrence J. Sejnowski between 1983-1986, when he was at the Johns Hopkins University, `Analyzing Cooperative Computation` (with Geoffrey E. Hinton), `Optimal Perceptual Inference` (with Geoffrey E. Hinton), `Boltzmann Machines : Constraint Satisfaction Networks That Learn` (with Geoffrey E. Hinton and David H. Ackley) and `A Learning Algorithm for Boltzmann Machines` (with Geoffrey E. Hinton and David H. Ackley). It definitely comforting me to be able to connect to this interesting piece of history that much of the foundational ideas about neural nets were essentially invented in JHU - something that I have never found mentioned enough in my neighbourhood.

Possibly the single-most interesting idea in all of physics that had always captivated my imagination was the story of Large-N Matrix Models : a beautiful mathematical construction which is prima-facie a Quantum Field Theory but mysteriously enough it has time and again been shown to encode critical features of also being a gravitational theory and it has also been known to be a framework for addressing various deep questions in algebraic geometry. I so much wanted to pursue this subject in the physics department and yet that opportunity never arrived. Maybe deep down one of the things that got me motivated about deep-learning is its superficial resemblance to Matrix Models in physics - and I do hope that someday we will be able to figure out a robust connection between the two. Many a days in grad-school, I woke up in the morning feeling depressed about not being able to do this research. It didn't help that during my Ph.D., the storm of Jackiw-Teitelboim gravity started blowing over physics : which is yet another exciting avatar of Matrix Models. It took a lot of psychological effort on my part to not disband all my ongoing thesis work and dive into this development which was clearly looking to be so much more exciting! On my table, I have always had open a copy of

Marcos Marino's lectures on this topic - hoping if someday I could finally get to thinking about these beautiful questions.

Long back I used to research in Quantum Field Theory with Prof. Shiraz Minwalla and in retrospect I don't think I had been fully conscious of the "magic" of how he works and thinks. His abilities are legendary about being able to intuit the essential characteristics of the answer bypassing the apparent complexity of the equations. It was only during my Ph.D. when I started to think of research "24×7" that I finally began to realize all the intangible things that I had actually learnt from him - which went a lot beyond the mechanics of the homotopy based classification of instantons and solitons that he had taught me as an undergrad. Deeper I went into research more I realized as to how profoundly had Prof. Shiraz Minwalla moulded my psyche and approach towards research.

In this respect I need to also mention the profound influence that Prof. S. Ramanan had on me as an undergrad. I can trace back to say that my interest in research level mathematics essentially started with his evening classes whereby he taught a bunch of us undergrads (at the Chennai Mathematical Institute) from his renowned book "Global Calculus". In retrospect it seems incredible that he was able to devise a way to teach us beginning undergrads about sheaf theory and Dolbeault cohomology. Prof. S. Ramanan instilled in me a sense of adventure and courage about doing mathematics which continues to shape how I think!

On almost half the days I found it difficult to motivate myself to come to work. But everyday I put up this fight against depression to get myself to the institute. Probably being able to systematize this was one of the biggest struggles I "won" during my Ph.D. - that I figured out a mental gymnastics routine to be able to do my usual 10 AM-10PM schedule of work everyday despite an almost persistent feeling of sadness. This struggle against a perpetually imminent emotional collapse was far harder than the struggle with any mathematics question that I tried during my grad-school.

I often feel that maybe the only talent I have is my ability to get a certain amount of mathematics done despite these drowning feelings. The situation was essentially caused and continuously exacerbated by my increasing levels of academic loneliness and absence of shared interests with people around. Another factor that has been constantly gnawing at me from inside is that my current path has been causing a fast erosion of my previous geometric studies since nothing I do now is related to the kind of mathematics that I was pursuing before starting grad-school, like studying Riemann surfaces. I am hoping that sometime in future I shall be able to connect back to that path!

I have never really liked going to classes and sitting through courses. Hence it became a huge struggle to maintain my scientific enthusiasm against the tide of the usual rigmarole of grad-school that

seemed to be heavily weighing me down. I guess, I do not have any of the socially expected characteristics of being an "academic" and the process of Ph.D. has left me entirely conflicted between this realization and my intrinsic energy to relentlessly pursue the research questions that interest me. I do not yet know whether I can be a researcher or whether, if at all anywhere, I fit into the panoply of academia.

It is to be noted here as to how much my career was affected by Prof. Mauro Maggioni's course titled, "High-Dimensional Approximation, Probability, and Statistical Learning" It was definitely a turning point in my journey. I had come into grad school with neither any knowledge nor any inclination towards anything in probability theory or statistics. I had almost never found anything interesting about these subjects but as it turns out that its impossible to do deep-learning theory without these. This course by Prof. Mauro Maggioni immensely helped me see probability theory through a more familiar lens of geometry. This course gave me a psychological bridge between a form of thinking which was natural to me and the new terrain of probability theory which I had to get comfortable with to make progress with deep-learning. Prof. Mauro Maggioni introduced me to this book by Roman Vershynin, "High-Dimensional Probability" and that eventually became a pivot for all of my graduate studies. This book was a starting point from which I launched into reading other similar writings by Phillip Rigollet and Roman Handel that are getting increasingly read by contemporary researchers in this field. I went on to do all the courses that Prof. Mauro Maggioni ever offered!

In this context I need to express my immense gratitude towards Prof. Laurent Younes, not only for his careful reading of my thesis but also for checking some of my results even during the course of my studies. His readings have led to at least two major changes in my papers and no words can express how much I am grateful to him for these two significant contributions that he has made to my research.

The last two and a half years were essentially just me sitting in some corner of the institute scribbling away at questions which I could not get anyone I knew motivated about. I tried meeting anyone I thought who might have some overlap with stuff I was getting excited about but I essentially failed to convince almost everyone about the merits in my directions. This complete emotional failure eventually became the defining characteristic of my Ph.D. journey. It was a situation of complete despair and I never found a solution to it, except to learn to accept the desolation. I am very far from being a "scientist in the attic" and I think I thrive the most in collaborative atmospheres, while brainstorming freely with others for hours. Given this natural emotional disposition of mine, the process of getting a Ph.D. seemed to be happening in an atmosphere entirely antithetical to whatever feels natural or comfortable to me. Research seems to firstly be a challenge of survival in solitary

confinement. Almost everyday I have doubted if I could have continued doing this for even one more day unless something dramatically changed about this situation.

I am enormously grateful to two of my grad student peers, Akshay Rangamani and Ramchandran Muthukumar, who at various points for short periods of time had a lot of regular meetings with me and wrote papers with me. The process of brainstorming with them was one of the few things that kept me stable through those periods. Likewise Soham De (then a grad student at the University of Maryland, now at DeepMind, UK) and co-grad student Pushpendre Rastogi were two other incredible peers from the experimental deep-learning community who spent an enormous amount of time chasing down ideas with me. In the process they taught me a whole lot of things about how deep-nets behave - things I could have only learnt by poring through experimental data with them! There were very many others in JHU, discussions with whom have contributed to my thinking : Enayat Ullah, Prof. Raman Arora, Prof. Trac Tran, Prof. Rene Vidal, Joseph Paat, Prof. James Spall, Prof. Daniel Robinson and Prof. Jeremias Sulam.

Almost every month I have thought of changing my research direction to something more mainstream in the campus, in hope if doing so would get me more human interactions and discussions. But as much as I desperately struggled to find collaborations and brainstorming sessions with others in the campus, I couldn't ignore the fact that I was deeply attracted to the questions of deep-learning theory. The spectacular progress the subject was making in the world outside was what kept me inspired and yet it was this same attraction which made my loneliness entirely untenable. I personally feel that it was almost a miracle that I could continue doing some research despite the tremendous emotional strain I found myself in. For reasons not entirely clear to me, in this process I often came to appreciate the famous quote by Albert Camus, "The only way to deal with an unfree world is to become so absolutely free that your very existence is an act of rebellion."

It is hard to underestimate how much social media like Twitter and Facebook have contributed to deep-learning theory. These have phenomenally impacted the subject not just in terms of actually funding such research but also via the volume of high-quality discussions on these topics that happens between the experts on those platforms. Often my only way of getting to know what the most exciting directions in the community were, was via keeping track of the discussions on the Twitter threads between the luminaries in the field! This is essentially how I found my post-doc position with the renowned mathematician in the field, Prof. Weijie Su at Wharton, UPenn!

It was also through such social networks that I became aware of Prof. Dan Roy's works and thanks to his kindness I landed an internship with him at the Vector Institute of Artificial Intelligence at

UToronto. That was an eye-opening experience which not only got me the last chapter of this thesis but it also got me introduced to a high-voltage learning theory atmosphere in Vector - something quite unlike whatever I had ever experienced before! But maybe more than that, the couple of months I spent attending Prof. Dan Roy's group meetings got me introduced to this exciting subject of "Stochastic Differential Equations (SDEs)". It was a gust of fresh air, which in my ways fundamentally altered my research thinking. As far as I can forsee the interface between SDEs and deep-learning is going to be a focal point of my works in the coming months! In this context I need to express my immense gratitude for Prof. Avanti Athreya in the department who patiently answered many of my questions about SDEs and thus further propelled my interest in this field.

It was also though social media that I came to interact with seniors in the field like Prof. Animashree Anandkumar (Caltech) and Prof. Lalitha Sankar (ASU). I shall be immensely grateful for the amount of encouragement that Prof. Lalitha Sankar provided me in my journey through the last couple of years. Many a days were lit up by virtue of her kind words. Also an equal amount of thanks is to due to Kenji Kawaguchi (then a grad student at MIT and now a faculty at NUS), Prof. Chinmay Hegde (NYU) and Prof. Rajarshi Mukherjee (Harvard) who have had extensive brain-storming sessions with me on various deep-learning questions and I hope that some of the ideas from these meetings will see the light of the day in near future. It is also to be noted as to how much Prof. Kazi Rajibul Islam at UWaterloo inspired me to write an introductory article on deep-learning targeted towards undergrads and high-school students. He read and edited various drafts that I wrote for this purpose and eventually that became the chapter in my thesis that has been titled, "An Informal Introduction to Deep-Learning". Prof. Kazi Rajibul Islam also arranged to get it translated into Bangla by Dr. Amiya K. Maji at the Purdue University. They have in turn helped disseminate the translation into a wider audience via their web-portal Bigyan.org.in

Despite these sporadic sparks of inspiration from the outside world, eventually it became very hard to not think of graduate school like a "zero-sum game" - as understood colloquially. The emotional burnout and other losses in life seemed to compound so massively that ultimately it became hard for me to see any bigger point in the process than just wanting this degree which happens to be a qualification cut-off for certain jobs I would like to be able to get someday. A whole lot thanks is to due to co-grad student Aarushi Goel, who helped many a days by volunteering to play badminton with me or by just sitting around listening to me. This emotionally excruciating process of going through grad school has taken a lot of toll on my family members who have stood through thick and thin with me : my mother Dr. Suranjana Sur, my maternal grand-mother Mira Sur, my sister Subhalakshmi, my aunt Debanjana Bhattacharyya, my cousin sister Debadrita Bhattacharyya and my uncle Dibyendu Bhattacharyya.

As I end my Ph.D. journey I feel an increasing need to go back to my roots to rediscover the things that really got me started into doing science at all. I began my undergrad studies reading Richard Feynman's famous lectures in physics and as I submit this thesis I find increased resonance in these words of his - which probably I had not begun to appreciate untill now,

"We find that the statements of science are not of what is true and what is not true, but statements of what is known to different degrees of certainty. Every one of the concepts of science is on a scale graduated somewhere between, but at neither end of, absolute falsity or absolute truth."

- Richard Feynman

A special thanks is due to Zachary Lubberts and Sayan Chakraborty for helping me set up this LaTeX template for the thesis!

Contents

List of Tables

List of Figures

An Informal Introduction to Deep Learning

We who speak Bengali owe an infinite debt to the legendary Satyajit Ray and it goes well beyond him having given us the timeless movies that he created. Growing up in a typical Bengali household full of books (on almost every conceivable subject!), maybe for many of us our first idea of "artifical intelligence" can be traced to the friendly humanoids like Robu and Bidhushekhar which were created by Professor Shonku in the famous series of stories penned by Satyajit Ray. In retrospect it was indeed lucky that Professor Shonku happened in our lives much before we encountered the more ominous view of robots as was made famous by Isaac Asimov's "Three Laws of Robotics". Ofcourse even today in 2020 we are still nowhere close to what was imagined in Satyajit Ray's fiction but something has dramatically changed in the last $5-6$ years. In this chapter we will try to get a feel of this ongoing revolution while keeping the technical aspects low enough to be accessible within the scope of high-school science. We should note at the very outset that opinions remains widely divided about how we should perceive this recent upsurge and much of what we present here is obviously heavily coloured by the technical parts of the thesis that will follow this chapter.

Maybe many of the readers have probably heard of the recent spectacular successes of "machines" called AlphaZero in being able to play games like chess at unprecedented levels of proficiency. These successes have revealed structures and possible strategies about the game of chess which had never been seen before! But these forms of artificial intelligence (unfortunately!) do not look like Professor Shonku's robots. Turns out that anthropomorphism isnt of any particular advantage if we limit our notions of intelligence to such abilities as required to play difficult strategy games like chess or poker or being able to create new paintings which mimic the style of Vincent van Gogh or being able to fluently translate between multiple languages. In the last couple of years suddenly these have become possible to do in an automated way because of our new found ability to computationally leverage the power of what are called "Deep Neural Networks" or DNNs or "neural nets" or "neural networks" or sometimes just "nets". The myriad of ways in which we can "train" a DNN to perform human-like tasks are collectively called "Deep Learning"

It can be somewhat tedious to install on one's home computer the (freely available) softwares like TensorFlow or PyTorch and get a hands-on feel for the advanced applications of neural nets that were mentioned above. The developers of these softwares continue to make progress to make the installation processes increasingly easy so that more people can put this evolving technology to use. Such efforts have led to the creation of platforms like "Google Colab" where one can write codes to run small neural nets without having to install the full softwares. For immediate motivation let's see this incredibly beautiful (and mind-bogglingly surprising!) demonstration that is easily available on this website, https://thispersondoesnotexist.com/. Every time we refresh this page we will be shown a seemingly human photograph (which sometimes might have minor defects), just that this photograph is completely artificially generated by a neural network! In a sense this person is purely the net's imagination and he/she does not actually exist! So how did the net manage to "draw" such realistic human faces? This mechanism is still highly ill-understood and our best efforts at making sense of this involves the branch of mathematics called "Optimal Transport". This is the same field of research for which Cedric Villani got the Fields Medal in 2010. This esoteric mathematical idea of optimal transport has mysterious ramifications in the world of neural nets and we have possibly only barely scratched the surface of this interface.

Though applications like the one described above about artificial generation of human-like faces are the cutting-edge of applied research in neural nets, these are not the commonly used tests for theory. There are more standardized artificial intelligence tasks on which we have decades of benchmarks of performance and new techniques are often compared on those. One such task is of classifying images into meaningful categories when the neural net (or in general any candidate "machine") is input a high-dimensional vector representing the image. For comparison recall that its at about 9 months of age that a human baby first starts being able to match daily life objects to their photographs. But the nets are no match for babies! Babies can recognize a banana the next time even after having seen just a single banana once. Unfortunately our best nets still need to see a lot of bananas before learning to categorize it correctly when shown a new one! This human-machine gap is deeply mysterious and an emerging direction of research.

There are two common datasets of images which are used for this test namely the "CIFAR" (Canadian Institute For Advanced Research) database and the MNIST ("Modified National Institute of Standards and Technology") database. CIFAR dataset was created in 2009 by Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. It contains millions of low resolution images grouped into thousands of categories like birds, aeroplanes, cars etc. The task of the trained machine is to correctly predict the category when a randomly picked image from this set is input to the machine. In the figure below we have shown a sample of the MNIST dataset which contains images of hand-written digits from

0 to 9 and the task of the trained machine is to recognize the number correctly when shown a randomly picked handwritten digit from the set. This was introduced and explored in the seminal paper from 1998 called, "Gradient-Based Learning Applied to Document Recognition" written by some of the biggest stalwarts in the field, Y. LeCun, L. Bottou, Y. Bengio and P. Haffner. And even today we continue to use MNIST as a baseline for testing theory about classification tasks.

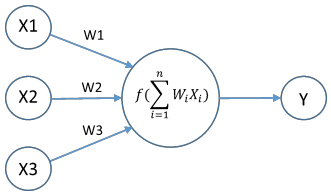
A small part of the famous MNIST database

Its worth pointing out that CIFAR is widely considered to be a much more difficult test than MNIST. There are fundamental questions about being able to mathematically justify this difference in difficulty and theory of this kind is still not fully developed. This brings us to a deep mystery that we hardly understand as to when is an artificial intelligence task easy and when is it difficult! Research is only beginning in this direction.

Now that we have seen some cutting-edge applications and methods of testing artificial intelligence let us focus on understanding the specific implementation of DNNs that we are interested in.

DNNs have existed in some form or the other since the 1958 work by a psychologist at the Cornell University named Frank Rosenblatt, who then called his idea "Perceptron". Many might say that our current way of thinking about neural nets comes from the famous 1986 paper titled ``Learning representations by back-propagating errors'' by David Rumelhart, Ronald Williams and Geoffrey Hinton. Geoffrey E. Hinton is often credited to be the pioneer of deep learning and interestingly Hinton too did his undergraduate studies in psychology! Its worth noting that despite the ideas having been there since decades, till very recently we could never actually get nets to do anything surprising in practice. This so-called "A.I winter" was finally ended in part by the recent dramatic developments in computer hardware. The ongoing artificial intelligence revolution can be said to have been ignited by the iconic 2012 paper from University of Toronto titled ``ImageNet Classification with Deep Convolutional Neural Networks'' by Alex Krizhevsky, Ilya Sutskever and Geoffrey E. Hinton. This showed that deep nets can be used to classify images into meaningful categories with almost human-like accuracies! Now lets try to understand what is the precise mathematical description of a DNN!

DNNs are what could be called "mathematical circuits". These can be thought of as a certain peculiar class of functions which are defined via diagrams which look like circuits. In school we get very familiar with the physics of electrical circuits - which carry electrical potential. Mathematical circuits are similar but instead their imagined wires carry algebraic instructions to multiply or add numbers to the input to the wire. We know that electrical circuits can be augmented to do more useful things by embedding inside them "non-linear" components like resistors, capacitors and inductors. There are called non-linear because the voltage drop across them is not a linear function of the current passing through them. Similarly these DNNs have embedded inside them gates each of which is designed to implement a certain "activation function" which is typically a non-linear function mapping the real line to itself. The following diagram represents an example of such a single gate and is thus one of the most elementary possible examples of a neural net.



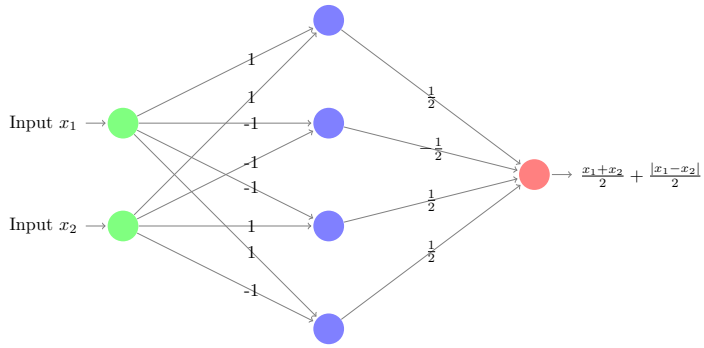
The above neural gate will be said to use $f : \mathbb{R} \to \mathbb{R}$ as the activation function, to create a map/function which takes as input any $3-$dimensional vector (x_1, x_2, x_3) and gives as output the real number, $f(w_1x_1 + w_2x_2 + w_3x_3)$. We think of the activation function at the gate f, to be getting as input the linear sum, $\sum_{i=1}^{3} w_i x_i$. These three real parameters above, w_1, w_2 and w_3 are called the 'weights'. Usually there are many wires coming out of the gate (instead of the single Y in the above) and in that case the gate is defined to pass on the same value to all of them.

As of today almost all implementations of DNNs use the "Rectified Linear Unit (ReLU)"

$$\text{ReLU} : \mathbb{R} \to \mathbb{R}$$
$$x \mapsto \max\{0, x\}$$

To develop more intuition lets use the building blocks above to construct a net with a few more gates which actually computes a familiar useful function.

In the above we see an example of a "1-DNN" i.e. a DNN with one layer of gates indicated in blue. The above neural network would be said to be of size 4 since it has 4 activation gates. Lets assume that the activation function at these blue gates is the ReLU function defined above. Then we claim



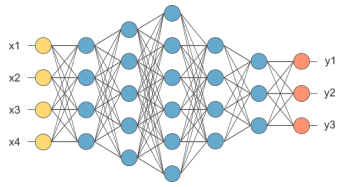
that the above circuit is computing the $\mathbb{R}^2 \to \mathbb{R}$ function given as $(x_1, x_2) \mapsto \max\{x_1, x_2\}$. But how do we convince ourselves that this is indeed what is happening? We can start with realizing that the top most blue activation gate is getting as input the number $x_1 + x_2$ and this can be inferred from the weights on its two incoming edges. (recall how the single gate was defined to operate in the diagram on the previous page) Further one can read off from the weight on the outgoing edge of this top most ReLU gate that it is passing on to the red output gate the number $\frac{1}{2}\max\{0, x_1 + x_2\}$. Thus if we carefully follow the computations happening on each edge and gate then we can conclude that the mathematical circuit/neural net above is indeed computing the maximum of its two inputs. Let's see a specific example for check : say $x_1 = 2$ and $x_2 = 5$. Then the blue ReLU gates are getting $7, -7, 3$ and -3 as inputs respectively, from the top most gate to the bottom most. These gates are then passing on to the red output gate the numbers, $\frac{7}{2}, -\frac{1}{2}\max\{0, -7\} = 0, \frac{1}{2}\max\{0, 3\} = \frac{3}{2}$ and $\frac{1}{2}\max\{0, -3\} = 0$. Finally the red output gate is adding these up to give as output $\frac{7}{2} + \frac{3}{2} = 5$ which is indeed $\max\{2, 5\}$.

Later in the thesis in Chapter 2 we will prove that to compute the maximum of n numbers atmost $\log(n)$ layers of activation are sufficient. In that same chapter we will study another useful class of neural functions which have been very important in theory building. For some real numbers $w > 0$ and $a > 0$ consider the function, $f(x) = \max\left\{0, \frac{1}{w} - \frac{1}{w^2} \times |x - a|\right\}$. Its easy to see that this is zero everywhere except on the interval $[a - w, a + w]$ where it rises up as a triangle peaked at $x = a$. Now that we have seen the max function example above, one can try to solve the fun puzzle of writing down a neural net with a single layer of activations which can represent this "triangle wave" function. With some more tricks we will see in Chapter 2 that one can try to create nets which will represent a wave form with multiple triangles. Interestingly its still unclear as to what is the appropriate analogue of these waves in high-dimensions!

These nets which compute the maximum of their input numbers and the triangle waves above often form the building blocks of how we think about the more complicated functions that the nets can compute. In general questions about the representation power of a specific circuit/network design

can be extremely difficult to answer and in some cases the answers have required the use of very sophisticated mathematics as we will see in Chapter 2. At the core of trying to explain cutting-edge mind-boggling experiments cited earlier like https://thispersondoesnotexist.com/, there lies in effect more advanced forms these kinds of questions about the function space of nets.

To see more advanced ideas we need to think more generally in terms of diagrams or "architectures" : an example of which is given below. (For readers familiar with graph theory one can imagine the underlying diagram to be that of a directed acyclic graph where all edges are pointing to the right.)



Unlike the previous two examples, in the above circuit no "weights" have been assigned to the edges of the above graph. So one should think of this diagram as representing the entire set of all the $\mathbb{R}^4 \to \mathbb{R}^3$ functions which can be computed by the above architecture for a *fixed* choice of "activation functions" (like, ReLU as defined above) at each of the blue nodes and for all possible values of weights/real-numbers that can be assigned to the edges. The 4 yellow nodes are where a $4-$ dimensional input vector will go in as input and the 3 orange nodes are where the $3-$dimensional output vector will come out. Unlike the previous diagrams where every edge carried a single real number, in general every edge can be assigned two real numbers/weights/parameters say (a, b). It is to be understood as specifying that when that edge gets a real number say x as input on its left end then it will give the number $ax + b$ as the output on its right end. Thus the total number of parameters specifying a neural function can be at most twice the number of edges. Some of the largest nets in operation today (called "AmoebaNet-D") have 600 million parameters. For comparison recall that the human brain has about $60-80$ billion neurons and each of them have about 10^4 synaptic connections to other neurons. This might motivate one to say that these nets which hope to model intelligence are still quite small in comparison to the human brain!

The above diagram would be said to represent a class of neural functions with 5 "layers" of activations, the blue nodes. We recall that the function we had seen earlier, $f(x_1, x) = \max\{x_1, x_2\}$ was such that it could be represented using just 1 layer of activation. In this context it is worth pointing out that we still don't know if say the maximum of 5 numbers can be computed using 2 layers of gates or does it necessarily need 3 such layers of gates!

Now that we have started to think in terms of architectures we end this chapter by pointing out that a very crucial unresolved issue here is to be able to understand, that of all the functions that can be

well *approximated* by a chosen architecture, how many of them show rapid oscillations - like imagine the triangle-wave that we saw earlier but with many (but finite) number of triangles.

This was just the tip of the iceberg and there is a lot more to this story. Lets get a quick glimpse of some of that! In deep-learning we will often assume that the actual artificial intelligence task that one desires to accomplish can be reformulated as trying to minimize some real valued function which is often called the "loss function". Our ever increasing experience is that with enough ingenuity one can often write the correct loss function - which will capture the original question as a function which maps the space of functions of a neural architecture (and available data) to non-negative real numbers. And as one might expect we try to minimize the loss function over the space of weights of the net (and hence over the space of functions represented by the given architecture) by approximately moving along the local gradients of the loss function. Thus it is immensely critical that we choose the right architecture - and this is currently almost a form of art! Research is only beginning in this direction of finding systematic methods for making a good choice of architecture. Even after an architecture has been chosen we are faced with the massive question of actually doing this search through its space of functions/weights of the net to find the minimum of the loss.

In the description above we have so far hidden an immense complication which we now necessarily need to confront - that in actual practice information about this loss function is often only partially known! In general this is a very complicated question about searching for an optimal function in a function space while being guided only by a crude estimate of the true optimality criteria. This brings us to the vast field of "stochastic optimization" - and we will see many provable avatars of this in this thesis. We hope that the appetite of the reader has been adequately whetted and at this point they might want to read some of the recently released books on deep-learning like these two freely available beautiful introductory books which give a magnificent overview of this exciting new subject https://www.deeplearningbook.org/ and d2l.ai.

Chapter 1

A Summary of the Results in This Thesis

Deep learning has brought about a paradigm shift in our quest for general artificial intelligence (LeCun, Bengio, and Hinton, 2015). Powered by concurrent technological advances neural nets have in recent times beaten all previous benchmarks in playing hard strategy games like chess and Go, (Silver et al., 2017; Silver et al., 2018) and have also radically pushed forward the technology towards self-driving cars (Fridman et al., 2017). But on the other hand the methods employed to make deep learning practical remain highly mysterious and challenging to prove guarantees about. During my PhD. I have been extremely passionate about figuring out mathematically rigorous ways to understand deep-learning. We begin to give a summary of the results obtained by first setting up the mathematical notation needed to talk about nets.

1.1 Defining Deep Neural Nets

The crucial component that goes into defining a neural net is the "activation function", often denoted as σ. Historically the σ that was in vogue at the beginning of the subject was the "sigmoid function", $\mathbb{R} \ni x \mapsto \sigma(x) = \frac{1}{1+e^{-\lambda x}}$ for some $\lambda > 0$. But for almost all applications of neural nets today it seems that the most widely used activation function is the "Rectified Linear Unit (ReLU)"

$$\begin{aligned} \text{ReLU} : \mathbb{R} &\to \mathbb{R} \\ x &\mapsto \max\{0, x\} \end{aligned}$$

In standard practice the notion of ReLU is overloaded to denote the following function operating entrywise, $\text{ReLU} : \mathbb{R}^n \ni \mathbf{x} \mapsto (\max\{0, x_1\}, \max\{0, x_2\}, \ldots, \max\{0, x_n\}) \in \mathbb{R}^n$

Definition 1. [ReLU DNNs] Given $k, w_0, w_1, w_2, \ldots, w_k, w_{k+1} \in \mathbb{N}$, one defines a *depth* $k+1$ "ReLU Deep Neural Net (DNN)" as the following function,

$$\mathbb{R}^{w_0} \ni \mathbf{x} \mapsto f(\mathbf{x}) = \mathrm{A}_{k+1} \circ \mathrm{ReLU} \circ \mathrm{A}_k \circ \cdots \circ \mathrm{A}_2 \circ \mathrm{ReLU} \circ \mathrm{A}_1 \in \mathbb{R}^{\mathbf{w}_{k+1}} \tag{1.1}$$

where $\mathrm{A}_i : \mathbb{R}^{w_{i-1}} \to \mathbb{R}^{w_i}$ for $i = 1, \ldots, k+1$ is a set of $k+1$ affine transformations. The positive integers $w_1, \ldots, w_k$ are said to specify the *widths* of the *hidden layers* or *layers of activation*. The number $\max\{w_1, \ldots, w_k\}$ is called the *width* of this ReLU DNN. The *size* of the ReLU DNN is defined to be the number of univariate activation gates used and that can be easily seen to be $w_1 + w_2 + \ldots + w_k$.

Such a ReLU DNN is sometimes also called a $(k+1)$-layer ReLU DNN, and is said to have k *hidden layers*. Number of layers or depth of the net can be seen to be measuring as the length of the shortest path from the input to the output of the directed acyclic graph that naturally represents such a neural network - of which we have already seen examples in the previous chapter on informal summary and we see another in Figure 1.1 given below. □

For any $(m, n) \in \mathbb{N}$, let $\mathscr{A}_m^n$ adenote the class of affine and affine transformations from $\mathbb{R}^m \to \mathbb{R}^n$, respectively. Thus we introduce a compact notation for the class of width specified DNNs as follows,

Definition 2. We denote the class of $\mathbb{R}^{w_0} \to \mathbb{R}^{w_{k+1}}$ ReLU DNNs with k hidden layers of widths $\{w_i\}_{i=1}^k$ by $\mathscr{F}_{\{w_i\}_{i=0}^{k+1}}$, i.e.

$$\mathscr{F}_{\{w_i\}_{i=0}^{k+1}} := \{\mathrm{A}_{k+1} \circ \mathrm{ReLU} \circ \mathrm{A}_k \circ \cdots \circ \mathrm{A}_2 \circ \mathrm{ReLU} \circ \mathrm{A}_1 \mid \mathrm{A}_i \in \mathscr{A}_{w_{i-1}}^{w_i} \forall i \in \{1, \ldots, k+1\}\} \tag{1.2}$$

Corresponding to any affine transformation A_i above we will typically decompose its action via a linear transformation (*"weight matrix"*) W_i and a vector $\mathbf{b}_i$ s.t $\mathbf{x} \mapsto \mathrm{A}_i\mathbf{x} = \mathrm{W}_i\mathbf{x} + \mathbf{b}_i$. This is particularly helpful in setting up the notation for a particular class of neural nets "Autoencoders" that we shall often consider in this thesis and which we specify below.

1.1.1 Notation for a special class of Autoencoders

Let $\mathbf{y} \in \mathbb{R}^n$ be the input vector to the autoencoder, $\{\mathrm{W}_i\}_{i=1,..,\ell}$ denote the weight matrices of the net and $\{\mathbf{b}_i\}_{i=1,..,2\ell}$ be the bias vectors. Then the output $\hat{\mathbf{y}} \in \mathbb{R}^n$ of the autoencoder (mapping $\mathbb{R}^n \to \mathbb{R}^n$) is defined as,

$$\hat{\mathbf{y}} = \mathrm{W}_1^\top \sigma(\dots \sigma(\mathrm{W}_{\ell-1}^\top \sigma(\mathrm{W}_\ell^\top \mathbf{a} + \mathbf{b}_{\ell+1}) + \mathbf{b}_{\ell+2}) \dots) + \mathbf{b}_{2\ell}$$

where

$$\mathbf{a} = \sigma(\mathrm{W}_\ell \sigma(\dots \sigma(\mathrm{W}_2 \sigma(\mathrm{W}_1 \mathbf{y} + \mathbf{b}_1) + \mathbf{b}_2) \dots) + \mathbf{b}_\ell)$$

This defines an autoencoder with $2\ell - 1$ hidden layers using the ℓ weight matrices and the 2ℓ bias vectors defined above. The particular symmetry that has been imposed among the layers leading up to the $\mathbf{a}$ and those that act on $\mathbf{a}$ is what leads to this arrangement being called *"weight tied"*. Such autoencoders are a fairly standard setup that have been used in previous work (Arpit et al., 2015; Baldi, 2012; Kuchaiev and Ginsburg, 2017; Vincent et al., 2010).

A special case of the above that we shall focus on is when $\ell = 1$ i.e its a weight tied autoencoder of depth 2 and $\mathbf{b}_2 = 0$. We shall use $\mathrm{W} = \mathrm{W}_1$, $\mathbf{b}_1 = \epsilon \in \mathbb{R}^h$ where h is the width of the net and the number of activation units used. Denoting the output of the hidden layer of activations as $\mathbf{r} \in \mathbb{R}^h$ we have for this case,

$$\hat{\mathbf{y}} = \mathrm{W}^T \mathbf{r} \text{ where } \mathbf{r} = \mathrm{ReLU}\left(\mathrm{W}\mathbf{y} - \epsilon\right) \tag{1.3}$$

We shall define the columns of $\mathrm{W}^\top$ (rows of W) as $\{\mathrm{W}_i\}_{i=1}^h$. A pictorial representation of such a depth 2 autoencoder is as given in Figure 1.1.

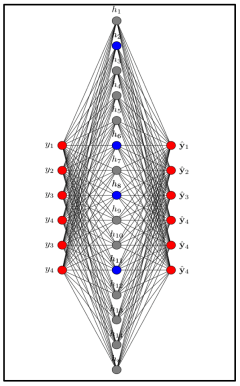


FIGURE 1.1: The above is the circuit representation of a depth 2, width 15 autoencoder mapping, $\mathbb{R}^4 \ni y \mapsto \hat{y} \in \mathbb{R}^4$

Deep learning, refers to a suite of computational techniques that have been developed recently for training DNNs. It started with the work of Hinton, Osindero, and Teh, 2006 (deep belief networks) and Salakhutdinov and Hinton, 2009 (deep Boltzmann machines) which gave empirical evidence that if deep architectures are initialized properly (for instance, using unsupervised pre-training), then we can find good solutions in a reasonable amount of runtime. This work was soon followed by a series of early successes of deep learning at significantly improving the state-of-the-art AI systems in speech recognition, image classification and natural language processing based on deep neural nets (Hinton et al., 2012; Dahl, Sainath, and Hinton, 2013; Krizhevsky, Sutskever, and Hinton, 2012; Le, 2013; Sutskever, Vinyals, and Le, 2014). While there is less of evidence now that pre-training actually helps, several other solutions have since been put forth to address the issue of efficiently training DNNs. These include heuristics such as dropouts (Srivastava et al., 2014), but also considering alternate deep architectures such as convolutional neural networks (Sermanet et al., 2014)

One of the fascinating aspects of trying to build a theory for deep-learning is that if we view this project through the lens of optimization theory then its setup is essentially opposite to how theory of optimization is studied in standard textbooks and courses. Typically one starts off with a well defined optimization problem (like Conic Programming) and then one studies the properties of its optima and the algorithmic aspects of solving it. But deep-learning has developed entirely on the sturdy shoulders of thousands of highly innovative experimenters who have caused this artificial intelligence revolution by developing a vast array of mysterious heuristics which work to get the neural net to perform tasks which would be "human like". To give an obvious example : there is no unambiguous way to quantify the fact that state-of-the-art GAN outputs look like realistic images but this is exactly the criteria we would want to use to judge whether a GAN has been trained well or not! As a subject deep-learning is predominantly defined by wildly successful algorithmic heuristics and often it's entirely unclear as to how the obviously wonderful performance of the trained net can be described as finding good solutions of some optimization problem! For some of the most exotic applications of neural nets, the debates continue to happen about what is the right optimization problem whose solution would correctly capture the success of the net.

But if we can agree about the "loss function (say ℓ)" to be used then at least for the most ordinary use cases the challenge of modern deep learning can be abstracted out as a particularly hard case of usual "learning theory". In such benign situations we can focus on wanting to solve the following function optimization/"risk minimization" question,

$$\min_{\mathbf{N} \in \mathcal{N}} \mathbb{E}_{\mathbf{z} \in \mathcal{D}} [\ell(\mathbf{N}, \mathbf{z})] \tag{1.4}$$

where ℓ is some lower-bounded non-negative function, members of $\mathcal{N}$ are continuous piecewise linear functions representable by some chosen neural net architecture and we only have sample access to the distribution $\mathcal{D}$. This reduces to the "empirical risk minimization" question when this $\mathcal{D}$ is an uniform distribution on a finite set of points. In the light of the previous discussion, the research results presented in this thesis can be seen to be focused on the following 3 critical aspects, (a) understanding mathematical properties of the neural function spaces on which this risk minimization is being attempted (Section 1.2), (b) proving guarantees about algorithms which can be used to approximately solve this question of neural risk minimization (Section 1.3) and (c) understanding the structure of the nets which are solutions to such risk minimization questions (Section 1.4)

1.2 Understanding the space of neural functions

In Chapter 2 we provide 3 main kinds of insights about the nature of neural functions. Firstly, we extend the recently published results in Telgarsky, 2016a to show that for every $k \in \mathbb{Z}^+$ there exists a *continuum* of hard functions which require $O(k^3)$ size to represent at depths $1 + k^2$ but will require $\Omega(k^k)$ (super-exponential in depth) size to approximate at depth $1 + k$. We also show that a kind of polytopes, called "zonotopes" have a natural relationship to neural nets i.e the ReLU nets can represent the gauge function of zonotopes and this in turn gives us an explicit construction of a *continuum* of ReLU functions with the largest number of affine pieces for large classes of architectures.

Secondly, we were intrigued by the question of finding non-trivial upperbounds on the run-time of algorithms which can find the *exact global minima* of empirical risk. We show how a collection of convex programming subroutines can be used to get algorithms for exact empirical risk minimization in depth 2 which run in $\text{poly}(\text{data})$ time at a fixed depth. Such faster-than-brute-force exact optimization algorithms remain unknown for higher depths. (Recently there has been a very interesting complexity theoretic paper from Berkeley, Manurangsi and Reichman, 2018 which builds further on this algorithm of ours.)

Lastly, we also investigate depth hierarchy theorems for ReLU nets (ending in a "Linear Threshold Function" (LTF) gate which maps $\mathbb{R} \ni y \mapsto -1 + 2\mathbf{1}_{y \geq 0} \in \mathbb{R}$) trying to compute Boolean functions. Many of the key results in this direction were achieved by extending to ReLU nets a method of random restrictions recently developed by Daniel Kane and Ryan Williams. This line of investigation has thrown up a lot of puzzling open questions about whether or not ReLU nets are more efficient at representing Boolean functions than usual Boolean circuits.

1.3 Landscape of neural nets and deep-learning algorithms

This theme is what can be said to be the mainstay of this thesis and it spans across 3 chapters.

In Chapter 3 we show 2 kinds of insights about training a ReLU gate. *Firstly* we give a very simple iterative stochastic algorithm to recover the underlying parameter $\mathbf{w}_*$ of the ReLU gate when realizable data allowed to be sampled online is of the form $(\mathbf{x}, \max\{0, \mathbf{w}_*^\top \mathbf{x}\})$. Compared to all previous such attempts the distributional condition we use is very mild, which essentially just captures the intuition that enough of our samples have to be such that $\mathbf{w}_*^\top \mathbf{x} > 0$.

Secondly we give an argument which establishes a first-of-its-kind mathematical control on the behaviour of gradient descent (with deliberate injection of noise) on the squared loss function of a single ReLU gate. It is to be noted that this argument doesn't need any distributional assumption beyond realizability of the labels and thus it makes us optimistic that this is a potentially interesting step towards explaining the success of this ubiquitously used heuristic. The key idea here is that of "coupling" which shows that from the iterates of noise injected gradient descent on the squared loss of a ReLU gate one can create a discrete super-martingale.

In Chapter 4 we focus on autoencoders and make progress about explaining their success. We were particularly inspired by the experimental works of Brendan Frey and Alireza Makhzani. We checked that actually an off-the-shelf RMSProp algorithm very easily do reasonably good autoencoding on MNIST even at depth 2. This piqued our interest to understand this better and we analyzed the landscape of the autoencoder under the usual sparse-coding generative model. Via a very elaborate analysis we are able to estimate the value of the gradient of the squared loss on depth 2 autoencoders whose input/output dimension is the same as that of the observed vectors in sparse-coding and the width of the network is the same as the sparse-code dimension.

This intricate analysis leads to the insight that the norm of this gradient decreases in a small neighbourhood of the original (unknown) dictionary as the sparse-code dimension increases. Such a proof of asymptotic criticality around the dictionary takes a step towards explaining why neural nets should be able to do dictionary learning. Works like Nguyen, Wong, and Hegde, 2019 have recently built on top of our analysis framework to show trainability proofs for autoencoders.

In Chapter 5 we focus on understanding the specific adaptive gradient algorithms, RMSProp and ADAM, which are implemented widely across almost all deep-learning tasks and are known to be the state-of-the-art in almost every application. We give the first ever proofs that (deterministic) RMSPRop and (deterministic) ADAM converge to criticality for smooth objectives without the assumption of convexity. We also motivate a class of first order moment constrained oracles in the

presence of which we can show the first ever proof of convergence of stochastic RMSProp with no convexity assumptions and at the same speed as SGD on convex functions.

We emphasize that this is particularly exciting in the context of recent results Reddi, Kale, and Kumar, 2018 which have shown that under the same setting of constant hyperparameter values ADAM used as an online optimizer cannot always get asymptotically zero average regret. We also shown extensive experiments on VGG-9 running on CIFAR-10 and across various sizes of autoencoders running on MNIST that ADAM's performance gets a consistent and curious boost (and thus it outperforms its competitors) when its β_1 (the parameter that controls the influence of the history of grdients on the current update) is pushed closer to 1 than its usual settings.

1.4 Estimating the risk function of neural nets

The long standing open-question in deep-learning is to be able to theoretically explain as to when neural nets which are massively over-parameterized happen to be high-quality solutions of the risk minimization problem defined in 1.4 - even when they fit the training data arbitrarily accurately. In recent times it has been increasingly realized that good risk bounds possibly necessarily need to depend on the training algorithm as well as the training data. The currently available methods to bound the risk function have been beautifully reviewed in this paper (Audibert and Bousquet, 2007). Here the authors have clubbed the techniques into primarly four categories, (1) "Supremum Bounds" (like generic chaining, Dudley integral, Rademacher complexity), (2) "'Variance Localized Bounds", (3) "Data-Dependent Bounds" and (4) "Algorithm Dependent Complexity". The last category includes PAC-Bayes bounds which have risen to prominence in recent times and is the crux of our most recently completed work described in Chapter 6

Rademacher complexity based bounds like Golowich, Rakhlin, and Shamir, 2018 and Bartlett, Foster, and Telgarsky, 2017 fail to give non-vacuous bounds when evaluated on the gigantic neural nets used in practice. In the PAC-Bayesian framework we slightly move away from trying to bound the risk and instead we try to bound an instance of "stochastic risk" which can be thought of as allowing for the neural net's weights/parameters to be noisy. This can be argued to be the most natural quantity to bound given that all successful neural training algorithms are stochastic and hence the trained net obtained from it is essentially a sample from a distribution on the neural function space induced by the training algorithm. By doing this shift in viewpoint, recent works like Dziugaite and Roy, 2017 and Zhou et al., 2018b have shown for the very first time that PAC-Bayesian bounds can give non-trivial risk bounds for practical neural nets. But the above bounds are "computational" in the sense that obtaining them requires an algorithmic search over a certain parametric space of distributions.

These experiments strongly motivate our current work seeking rigorous theoretical exploration of the power of PAC-Bayesian technology in explaining the learning ability of neural nets.

Previous PAC-Bayes bounds have used data dependent priors on the geometric mean of the spectral norms of the layer matrices to try to track the distance in the parameter space of the trained net from a fixed point in that space. In our work the first key idea we initiate is to track the distance of the trained net from its initialization by looking at two independent quantities (a) a non-compact parameter : the change from initialization of the norm of the vector of weights of the net (i.e the sum of Frobenius norms of the layer matrices for a net without bias weights) and (b) a compact part : the angular deflection of this vector of weights from initialization to the end of training. In this work we instantiate an elaborate mechanism of putting a two indexed grid of priors which can simultaneously be sensitive to both the above properties of the neural net training process.

Our second key idea is to realize that in the PAC-Bayesian framework one can leverage more out of the angle parameter by also simultaneously training a cluster of nets which are initialized close to the original net. Because of this use of clusters, compared to previous bounds our dependency on the distance from initialization is not only more intricate but we are also able to get more sensitive to the average case behaviour.

Compared to previous theories in this direction, (Neyshabur et al., 2017) we build into the formalism a larger number of data-dependent (and hence tunable) parameters. As a consequence we get a risk bound on nets which is empirically not only seen to be tighter than Neyshabur et al., 2017 but also has better/lower "rates" of dependency on the neural architectural parameters like depth and width.

We emphasize that the aforesaid ability to leverage the use of clusters of nets in tandem is critically hinged on us being able to prove methods of creating multi-parameter families of mixture of Gaussian distributions such that the given neural function remains stable when its weights are perturbed by noise sampled from these distributions. There are potentially far reaching implications of such theorems because of the intricate arguments made in recent times which motivate why finding provable compression algorithms for any class of nets is tied to being able to prove the existence of noise distributions to which this same class is resilient.

We go on to demonstrate two kinds of insights in our experiments. Over synthetic data and standard tests like CIFAR-10 we show that our bound performs consistently better than existing PAC-Bayesian bounds. Next we show in the experiments that the two parameters said above have a lot more structure than what theory is currently capable of leveraging. We observe in our experiments that the $2-$norm of the weight vector described above always undergoes a slight dilation during the training.

We also demonstrate that the angular deflection is predominantly determined by the underlying data-set/data-distribution and is only very slightly affected by the architecture of the net.

Current wisdom in the field suggests that observations like above about systematic behaviours of neural net training can potentially be leveraged into increasingly creative risk bounds for nets. Thus these experiments pave the way for our continuing exploration of even better bounds which can eventually lead to principled methods of choosing the right net to use for a given artificial intelligence task at hand.

Chapter 2

Exploring the Space of Neural Functions

2.1 Introduction

Neural networks with a single hidden layer of finite size can approximate any continuous function on a compact subset of $\mathbb{R}^n$ arbitrary well. This universal approximation result was first given for sigmoidal activation function in Cybenko, 1989, and later generalized by Hornik to an arbitrary bounded and non-constant activation function (Hornik, 1991) (and in turn it applied to ReLU nets as well). Furthermore, neural networks ending in a LTF gate have finite VC dimension (depending polynomially on the number of edges in the network), and therefore, are PAC (Probably Approximately Correct) learnable using a sample of size that is polynomial in the size of the networks (Anthony and Bartlett, 1999). However, neural networks based methods were shown to be computationally hard to learn (Anthony and Bartlett, 1999) and had mixed empirical success. Consequently, DNNs fell out of favor by the late 90s.

In this chapter, we formally study deep neural networks with rectified linear units; we refer to these deep architectures as ReLU DNNs. Our work is inspired by these recent attempts to understand the reason behind the successes of deep learning, both in terms of the structure of the functions represented by DNNs, (Telgarsky, 2015; Telgarsky, 2016b; Kane and Williams, 2015; Shamir, 2016), as well as efforts which have tried to understand the non-convex nature of the training problem of DNNs better (Kawaguchi, 2016; Haeffele and Vidal, 2015). Our investigation of the function space represented by ReLU DNNs also takes inspiration from the classical theory of circuit complexity; we refer the reader to Arora and Barak, 2009; Shpilka and Yehudayoff, 2010; Jukna, 2012; Saptharishi, 2014; Allender, 1998 for various surveys of this deep and fascinating field. In particular, our gap results are inspired by results like the ones by Hastad Hastad, 1986, Razborov Razborov, 1987 and Smolensky Smolensky, 1987 which show a strict separation of complexity classes. We make progress towards similar statements with deep neural nets with ReLU activation.

2.1.1 Notation and Definitions

Definition 3. [Piecewise linear functions] We say a function $f : \mathbb{R}^n \to \mathbb{R}$ is *continuous piecewise linear (PWL)* if there exists a *finite* set of polyhedra whose union is $\mathbb{R}^n$, and f is affine linear over each polyhedron (note that the definition automatically implies continuity of the function because the affine regions are closed and cover $\mathbb{R}^n$, and affine functions are continuous). The *number of pieces of* f is the number of maximal connected subsets of $\mathbb{R}^n$ over which f is affine linear (which is finite).

Many of our important statements will be phrased in terms of the following simplex.

Definition 4. Let $M > 0$ be any positive real number and $p \geq 1$ be any natural number. Define the following set:

$$\Delta_M^p := \{\mathbf{x} \in \mathbb{R}^p : 0 < \mathbf{x}_1 < \mathbf{x}_2 < \ldots < \mathbf{x}_p < M\}.$$

2.2 Exact characterization of function class represented by ReLU DNNs

One of the main advantages of DNNs is their representational ability. In this section, we give an exact characterization of the functions representable by ReLU DNNs. Moreover, we show how structural properties of ReLU DNNs, specifically their depth and width, affects their expressive power. It is clear from definition that any function from $\mathbb{R}^n \to \mathbb{R}$ represented by a ReLU DNN is a continuous piecewise linear (PWL) function. In what follows, we show that the converse is also true, that is any PWL function is representable by a ReLU DNN. In particular, the following theorem establishes a one-to-one correspondence between the class of ReLU DNNs and PWL functions.

Theorem 2.2.1. Every $\mathbb{R}^n \to \mathbb{R}$ ReLU DNN represents a piecewise linear function, and every piecewise linear function $\mathbb{R}^n \to \mathbb{R}$ can be represented by a ReLU DNN with at most $\lceil \log_2(n+1) \rceil + 1$ depth.

Proof Sketch: It is clear that any function represented by a ReLU DNN is a PWL function. To see the converse, we first note that any PWL function can be represented as a linear combination of piecewise linear convex functions. More formally, by Theorem 1 in (Wang and Sun, 2005), for every piecewise linear function $f : \mathbb{R}^n \to \mathbb{R}$, there exists a finite set of affine linear functions $\ell_1, \ldots, \ell_k$ and subsets $S_1, \ldots, S_p \subseteq \{1, \ldots, k\}$ (not necessarily disjoint) where each S_i is of cardinality at most $n+1$, such

that

$$f = \sum_{j=1}^{p} s_j \left(\max_{i \in S_j} \ell_i \right), \tag{2.1}$$

where $s_j \in \{-1, +1\}$ for all $j = 1, \ldots, p$. Since a function of the form $\max_{i \in S_j} \ell_i$ is a piecewise linear convex function with at most $n+1$ pieces (because $|S_j| \leq n+1$), Equation (2.1) says that any continuous piecewise linear function (not necessarily convex) can be obtained as a linear combination of piecewise linear convex functions each of which has at most $n+1$ affine pieces. Furthermore, Lemmas 2.A.2, 2.A.3 and 2.A.4 in the Appendix, show that composition, addition, and pointwise maximum of PWL functions are also representable by ReLU DNNs. In particular, in Lemma 2.A.4 we note that $\max\{x, y\} = \frac{x+y}{2} + \frac{|x-y|}{2}$ is implementable by a two layer ReLU network and use this construction in an inductive manner to show that maximum of $n+1$ numbers can be computed using a ReLU DNN with depth at most $\lceil \log_2(n+1) \rceil$.

While Theorem 2.2.1 gives an upper bound on the depth of the networks needed to represent all continuous piecewise linear functions on $\mathbb{R}^n$, it does not give any tight bounds on the *size* of the networks that are needed to represent a given piecewise linear function. For $n = 1$, we give tight bounds on size as follows:

Theorem 2.2.2. Given any piecewise linear function $\mathbb{R} \to \mathbb{R}$ with p pieces there exists a 2-layer DNN with at most p nodes that can represent f. Moreover, any 2-layer DNN that represents f has size at least $p - 1$.

Finally, the main result of this section follows from Theorem 2.2.1, and well-known facts that the piecewise linear functions are dense in the family of compactly supported continuous functions and the family of compactly supported continuous functions are dense in $L^q(\mathbb{R}^n)$ (Royden and Fitzpatrick, 2010)). Recall that $L^q(\mathbb{R}^n)$ is the space of Lebesgue integrable functions f such that $\int |f|^q d\mu < \infty$, where μ is the Lebesgue measure on $\mathbb{R}^n$ (see Royden Royden and Fitzpatrick, 2010).

Theorem 2.2.3. Every function in $L^q(\mathbb{R}^n)$, $(1 \leq q \leq \infty)$ can be arbitrarily well-approximated in the L^q norm (which for a function f is given by $||f||_q = (\int |f|^q)^{1/q}$) by a ReLU DNN function with at most $\lceil \log_2(n+1) \rceil$ hidden layers. Moreover, for $n = 1$, any such L^q function can be arbitrarily well-approximated by a 2-layer DNN, with tight bounds on the size of such a DNN in terms of the approximation.

Proofs of Theorems 2.2.2 and 2.2.3 are provided in Appendix 2.A. We would like to remark that a weaker version of Theorem 2.2.1 was observed in Goodfellow et al., 2013, Proposition 4.1 (with no bound on the depth), along with a universal approximation theorem (Goodfellow et al., 2013, Theorem 4.3) similar to Theorem 2.2.3. The authors of Goodfellow et al., 2013 also used a previous result of Wang (Wang, 2004) for obtaining their result. In a subsequent work Boris Hanin (Hanin, 2017) has, among other things, found a width and depth upper bound for ReLU net representation of positive PWL functions on $[0,1]^n$. The width upperbound is n+3 for general positive PWL functions and $n+1$ for convex positive PWL functions. For convex positive PWL functions his depth upper bound is sharp if we disallow dead ReLUs.

2.3 Benefits of Depth

Success of deep learning has been largely attributed to the depth of the networks, i.e. number of successive affine transformations followed by nonlinearities, which is shown to be extracting hierarchical features from the data. In contrast, traditional machine learning frameworks including support vector machines, generalized linear models, and kernel machines can be seen as instances of shallow networks, where a linear transformation acts on a single layer of nonlinear feature extraction. In this section, we explore the importance of depth in ReLU DNNs. In particular, in Section 2.3.1, we provide a smoothly parametrized family of $\mathbb{R} \to \mathbb{R}$ "hard" functions representable by ReLU DNNs, which requires exponentially larger size for a shallower network to represent. Furthermore, in Section 2.3.2, we construct a continuum of $\mathbb{R}^n \to \mathbb{R}$ "hard" functions representable by ReLU DNNs, which to the best of our knowledge is the first explicit construction of ReLU DNN functions whose number of affine pieces grows exponentially with input dimension.

2.3.1 Circuit lower bounds for $\mathbb{R} \to \mathbb{R}$ ReLU DNNs

In this section, we are only concerned about $\mathbb{R} \to \mathbb{R}$ ReLU DNNs, i.e. both input and output dimensions are equal to one. The following theorem shows the depth-size trade-off in this setting.

Theorem 2.3.1. For every pair of natural numbers $k \geq 1$, $w \geq 2$, there exists a family of "hard" functions representable by a $\mathbb{R} \to \mathbb{R}$ $(k+1)$-layer ReLU DNN of width w such that if it is also representable by a $(k'+1)$-layer ReLU DNN for any $k' \leq k$, then this $(k'+1)$-layer ReLU DNN has size at least $\frac{1}{2}k'w^{\frac{k}{k'}} - 1$.

In fact our family of hard functions described above has a very intricate structure as stated below.

Theorem 2.3.2. For every $k \geq 1$, $w \geq 2$, every member of the family of hard functions in Theorem 2.3.1 has w^k pieces and this family can be parametrized by

$$\bigcup_{M>0} \underbrace{(\Delta_M^{w-1} \times \Delta_M^{w-1} \times \ldots \times \Delta_M^{w-1})}_{k \text{ times}}, \tag{2.2}$$

i.e., for every point in the set above, there exists a distinct function with the stated properties.

The following is an immediate corollary of Theorem 2.3.1 by choosing the parameters carefully.

Corollary 2.3.3. For every $k \in \mathbb{N}$ and $\epsilon > 0$, there is a family of functions defined on the real line such that every function f from this family can be represented by a $(k^{1+\epsilon})+1$-layer DNN with size $k^{2+\epsilon}$ and if f is represented by a $k+1$-layer DNN, then this DNN must have size at least $\frac{1}{2}k \cdot k^{k^\epsilon} - 1$. Moreover, this family can be parametrized as, $\cup_{M>0}\Delta_M^{k^{2+\epsilon}-1}$.

A particularly illuminative special case is obtained by setting $\epsilon = 1$ in Corollary 2.3.3:

Corollary 2.3.4. For every natural number $k \in \mathbb{N}$, there is a family of functions parameterized by the set $\cup_{M>0}\Delta_M^{k^3-1}$ such that any f from this family can be represented by a k^2+1-layer DNN with k^3 nodes, and every $k+1$-layer DNN that represents f needs at least $\frac{1}{2}k^{k+1} - 1$ nodes.

Towards proving the above two theorems we first need the following definition and lemma,

Definition 5. For $p \in \mathbb{N}$ and $\mathbf{a} \in \Delta_M^p$, we define a function $h_{\mathbf{a}} : \mathbb{R} \to \mathbb{R}$ which is piecewise linear over the segments $(-\infty, 0], [0, \mathbf{a}_1], [\mathbf{a}_1, \mathbf{a}_2], \ldots, [\mathbf{a}_p, M], [M, +\infty)$ defined as follows: $h_{\mathbf{a}}(x) = 0$ for all $x \leq 0$, $h_{\mathbf{a}}(\mathbf{a}_i) = M(i \mod 2)$, and $h_{\mathbf{a}}(M) = M - h_{\mathbf{a}}(\mathbf{a}_p)$ and for $x \geq M$, $h_{\mathbf{a}}(x)$ is a linear continuation of the piece over the interval $[\mathbf{a}_p, M]$. Note that the function has $p+2$ pieces, with the leftmost piece having slope 0. Furthermore, for $\mathbf{a}^1, \ldots, \mathbf{a}^k \in \Delta_M^p$, we denote the composition of the functions $h_{\mathbf{a}^1}, h_{\mathbf{a}^2}, \ldots, h_{\mathbf{a}^k}$ by

$$H_{\mathbf{a}^1,\ldots,\mathbf{a}^k} := h_{\mathbf{a}^k} \circ h_{\mathbf{a}^{k-1}} \circ \ldots \circ h_{\mathbf{a}^1}.$$

Lemma 2.3.5. For any $M > 0$, $p \in \mathbb{N}$, $k \in \mathbb{N}$ and $\mathbf{a}^1, \ldots, \mathbf{a}^k \in \Delta_M^p$, if we compose the functions $h_{\mathbf{a}^1}, h_{\mathbf{a}^2}, \ldots, h_{\mathbf{a}^k}$ the resulting function is a piecewise linear function with at most $(p+1)^k + 2$ pieces, i.e.,

$$H_{\mathbf{a}^1,\ldots,\mathbf{a}^k} := h_{\mathbf{a}^k} \circ h_{\mathbf{a}^{k-1}} \circ \ldots \circ h_{\mathbf{a}^1}$$

is piecewise linear with at most $(p+1)^k + 2$ pieces, with $(p+1)^k$ of these pieces in the range $[0, M]$ (see Figure 2.1). Moreover, in each piece in the range $[0, M]$, the function is affine with minimum value 0 and maximum value M.

Proof. Simple induction on k. □

Proof of Theorem 2.3.2. Given $k \geq 1$ and $w \geq 2$, choose any point

$$(\mathbf{a}^1, \ldots, \mathbf{a}^k) \in \bigcup_{M>0} \underbrace{(\Delta_M^{w-1} \times \Delta_M^{w-1} \times \ldots \times \Delta_M^{w-1})}_{k \text{ times}}.$$

By Definition 5, each $h_{\mathbf{a}^i}$, $i = 1, \ldots, k$ is a piecewise linear function with $w+1$ pieces and the leftmost piece having slope 0. Thus, by Corollary 2.A.1, each $h_{\mathbf{a}^i}$, $i = 1, \ldots, k$ can be represented by a 2-layer ReLU DNN with size w. Using Lemma 2.A.2, $H_{\mathbf{a}^1,\ldots,\mathbf{a}^k}$ can be represented by a $k+1$ layer DNN with size wk; in fact, each hidden layer has exactly w nodes. □

Proof of Theorem 2.3.1. Follows from Theorem 2.3.2 and Lemma 2.A.7. □

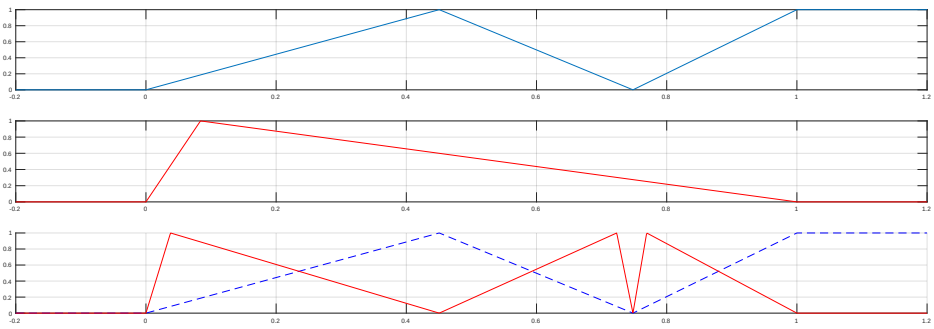


FIGURE 2.1: Top: $h_{\mathbf{a}^1}$ with $\mathbf{a}^1 \in \Delta_1^2$ with 3 pieces in the range $[0, 1]$. Middle: $h_{\mathbf{a}^2}$ with $\mathbf{a}^2 \in \Delta_1^1$ with 2 pieces in the range $[0, 1]$. Bottom: $H_{\mathbf{a}^1,\mathbf{a}^2} = h_{\mathbf{a}^2} \circ h_{\mathbf{a}^1}$ with $2 \cdot 3 = 6$ pieces in the range $[0, 1]$. The dotted line in the bottom panel corresponds to the function in the top panel. It shows that for every piece of the dotted graph, there is a full copy of the graph in the middle panel.

We can also get hardness of approximation versions of Theorem 2.3.1 and Corollaries 2.3.3 and 2.3.4, with the same gaps (upto constant terms), using the following theorem.

Theorem 2.3.6. For every $k \geq 1$, $w \geq 2$, there exists a function $f_{k,w}$ that can be represented by a $(k+1)$-layer ReLU DNN with w nodes in each layer, such that for all $\delta > 0$ and $k' \leq k$ the following holds:

$$\inf_{g \in \mathcal{G}_{k',\delta}} \int_{x=0}^{1} |f_{k,w}(x) - g(x)| dx > \delta,$$

where $\mathcal{G}_{k',\delta}$ is the family of functions representable by ReLU DNNs with depth at most $k'+1$, and size at most $k' \frac{w^{k/k'}(1-4\delta)^{1/k'}}{2^{1+1/k'}}$.

The depth-size trade-off results in Theorems 2.3.1, and 2.3.6 extend and improve Telgarsky's theorems from (Telgarsky, 2015; Telgarsky, 2016b) in the following three ways:

(i) If we use our Theorem 2.3.6 to the pair of neural nets considered by Telgarsky in Theorem 1.1 in Telgarsky, 2016b which are at depths k^3 (of size also scaling as k^3) and k then for this purpose of approximation in the ℓ_1-norm we would get a size lower bound for the shallower net which scales as $\Omega(2^{k^2})$ which is exponentially (in depth) larger than the lower bound of $\Omega(2^k)$ that Telgarsky can get for this scenario.

(ii) Telgarsky's family of hard functions is parameterized by a single natural number k. In contrast, we show that for every *pair* of natural numbers w and k, and a point from the set in equation 2.2, there exists a "hard" function which to be represented by a depth k' network would need a size of at least $w^{\frac{k}{k'}} k'$. With the extra flexibility of choosing the parameter w, for the purpose of showing gaps in representation ability of deep nets we can shows size lower bounds which are *super*-exponential in depth as explained in Corollaries 2.3.3 and 2.3.4.

(iii) A characteristic feature of the "hard" functions in Boolean circuit complexity is that they are usually a countable family of functions and not a "smooth" family of hard functions. In fact, in the last section of Telgarsky, 2015, Telgarsky states this as a "weakness" of the state-of-the-art results on "hard" functions for both Boolean circuit complexity and neural nets research. In contrast, we provide a smoothly parameterized family of "hard" functions in Section 2.3.1 (parametrized by the set in equation 2.2). Such a continuum of hard functions wasn't demonstrated before this work.

We point out that Telgarsky's results in (Telgarsky, 2016b) apply to deep neural nets with a host of different activation functions, whereas, our results are specifically for neural nets with rectified linear units. In this sense, Telgarsky's results from (Telgarsky, 2016b) are more general than our results in this paper, but with weaker gap guarantees. Eldan-Shamir (Shamir, 2016; Eldan and Shamir, 2016) show that there exists an $\mathbb{R}^n \to \mathbb{R}$ function that can be represented by a 3-layer DNN, that takes

exponential in n number of nodes to be approximated to within some constant by a 2-layer DNN. While their results are not immediately comparable with Telgarsky's or our results, it is an interesting open question to extend their results to a constant depth hierarchy statement analogous to the recent result of Rossman et al (Rossman, Servedio, and Tan, 2015). We also note that in last few years, there has been much effort in the community to show size lowerbounds on ReLU DNNs trying to approximate various classes of functions which are themselves not necessarily exactly representable by ReLU DNNs (Yarotsky, 2016; Liang and Srikant, 2016; Safran and Shamir, 2017).

Proof of Theorem 2.3.6. Given $k \geq 1$ and $w \geq 2$ define $q := w^k$ and $s_q := \underbrace{h_{\mathbf{a}} \circ h_{\mathbf{a}} \circ \ldots \circ h_{\mathbf{a}}}_{k \text{ times}}$ where $\mathbf{a} = (\frac{1}{w}, \frac{2}{w}, \ldots, \frac{w-1}{w}) \in \Delta_1^{q-1}$. Thus, s_q is representable by a ReLU DNN of width $w+1$ and depth $k+1$ by Lemma 2.A.2. In what follows, we want to give a lower bound on the ℓ^1 distance of s_q from any continuous p-piecewise linear comparator $g_p : \mathbb{R} \to \mathbb{R}$. The function s_q contains $\lfloor \frac{q}{2} \rfloor$ triangles of width $\frac{2}{q}$ and unit height. A p-piecewise linear function has $p-1$ breakpoints in the interval $[0,1]$. So that in at least $\lfloor \frac{w^k}{2} \rfloor - (p-1)$ triangles, g_p has to be affine. In the following we demonstrate that inside any triangle of s_q, any affine function will incur an ℓ^1 error of at least $\frac{1}{2w^k}$.

$$\begin{aligned}
\int_{x=\frac{2i}{w^k}}^{\frac{2i+2}{w^k}} |s_q(x) - g_p(x)| dx &= \int_{x=0}^{\frac{2}{w^k}} \left| s_q(x) - (y_1 + (x-0) \cdot \frac{y_2 - y_1}{\frac{2}{w^k} - 0}) \right| dx \\
&= \int_{x=0}^{\frac{1}{w^k}} \left| xw^k - y_1 - \frac{w^k x}{2}(y_2 - y_1) \right| dx + \int_{x=\frac{1}{w^k}}^{\frac{2}{w^k}} \left| 2 - xw^k - y_1 - \frac{w^k x}{2}(y_2 - y_1) \right| dx \\
&= \frac{1}{w^k} \int_{z=0}^{1} \left| z - y_1 - \frac{z}{2}(y_2 - y_1) \right| dz + \frac{1}{w^k} \int_{z=1}^{2} \left| 2 - z - y_1 - \frac{z}{2}(y_2 - y_1) \right| dz \\
&= \frac{1}{w^k} \left(-3 + y_1 + \frac{2y_1^2}{2 + y_1 - y_2} + y_2 + \frac{2(-2+y_1)^2}{2 - y_1 + y_2} \right)
\end{aligned}$$

The above integral attains its minimum of $\frac{1}{2w^k}$ at $y_1 = y_2 = \frac{1}{2}$. Putting together,

$$\|s_{w^k} - g_p\|_1 \geq \left(\lfloor \frac{w^k}{2} \rfloor - (p-1) \right) \cdot \frac{1}{2w^k} \geq \frac{w^k - 1 - 2(p-1)}{4w^k} = \frac{1}{4} - \frac{2p-1}{4w^k}$$

Thus, for any $\delta > 0$,

$$p \leq \frac{w^k - 4w^k\delta + 1}{2} \implies 2p - 1 \leq (\frac{1}{4} - \delta)4w^k \implies \frac{1}{4} - \frac{2p-1}{4w^k} \geq \delta \implies \|s_{w^k} - g_p\|_1 \geq \delta.$$

The result now follows from Lemma 2.A.7. □

2.3.2 A continuum of hard functions for $\mathbb{R}^n \to \mathbb{R}$ for $n \geq 2$

One measure of complexity of a family of $\mathbb{R}^n \to \mathbb{R}$ "hard" functions represented by ReLU DNNs is the asymptotics of the number of pieces as a function of dimension n, depth $k+1$ and size s of the ReLU DNNs. More precisely, suppose one has a family $\mathscr{H}$ of functions such that for every $n, k, w \in \mathbb{N}$ the family contains at least one $\mathbb{R}^n \to \mathbb{R}$ function representable by a ReLU DNN with depth at most $k+1$ and maximum width at most w. The following definition formalizes a notion of complexity for such a $\mathscr{H}$.

Definition 6 ($\text{comp}_{\mathscr{F}}(n,k,w)$)**.** The measure $\text{comp}_{\mathscr{F}}(n,k,w)$ is defined as the maximum number of pieces (see Definition 3) of a $\mathbb{R}^n \to \mathbb{R}$ function from $\mathscr{F}$ that can be represented by a ReLU DNN with depth at most $k+1$ and maximum width at most w.

Similar measures have been studied in previous works (Montufar et al., 2014; Pascanu, Montufar, and Bengio, 2013; Raghu et al., 2016). The best known families $\mathscr{F}$ are the ones from Theorem 4 of (Montufar et al., 2014) and a mild generalization of Theorem 1.1 of Telgarsky, 2016b to k layers of ReLU activations with width w; these constructions achieve $\left(\lfloor(\frac{w}{n})\rfloor\right)^{(k-1)n}(\sum_{j=0}^{n}\binom{w}{j})$and $\text{comp}_{\mathscr{F}}(n,k,s) = O(w^k)$, respectively. At the end of this section we would explain the precise sense in which we improve on these numbers. An analysis of this complexity measure is done using integer programming techniques in Serra, Tjandraatmadja, and Ramalingam, 2017.

Definition 7. Let $\mathbf{b}^1, \ldots, \mathbf{b}^m \in \mathbb{R}^n$. The zonotope formed by $\mathbf{b}^1, \ldots, \mathbf{b}^m \in \mathbb{R}^n$ is defined as

$$Z(\mathbf{b}^1, \ldots, \mathbf{b}^m) := \{\lambda_1 \mathbf{b}^1 + \ldots + \lambda_m \mathbf{b}^m : -1 \leq \lambda_i \leq 1, \;\; i = 1, \ldots, m\}.$$

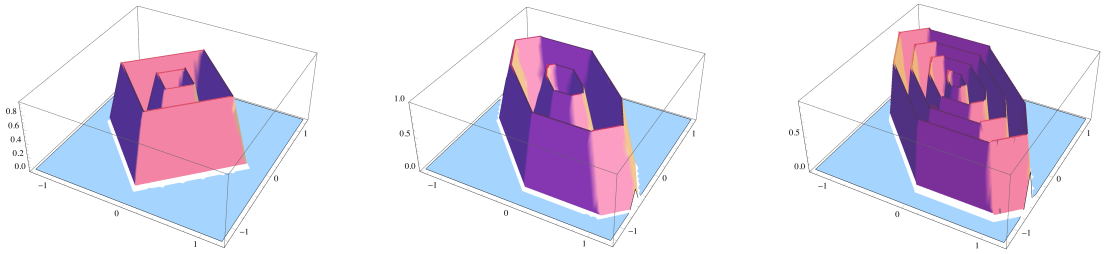
The set of vertices of $Z(\mathbf{b}^1, \ldots, \mathbf{b}^m)$ will be denoted by $\text{vert}(Z(\mathbf{b}^1, \ldots, \mathbf{b}^m))$. The *support function* $\gamma_{Z(\mathbf{b}^1,\ldots,\mathbf{b}^m)} : \mathbb{R}^n \to \mathbb{R}$ associated with the zonotope $Z(\mathbf{b}^1, \ldots, \mathbf{b}^m)$ is defined as

$$\gamma_{Z(\mathbf{b}^1,\ldots,\mathbf{b}^m)}(\mathbf{r}) = \max_{\mathbf{x} \in Z(\mathbf{b}^1,\ldots,\mathbf{b}^m)} \langle \mathbf{r}, \mathbf{x} \rangle.$$

The following results are well-known in the theory of zonotopes (Ziegler, 1995).

Theorem 2.3.7. The following are all true.

1. $|\text{vert}(Z(\mathbf{b}^1, \ldots, \mathbf{b}^m))| \leq \sum_{i=0}^{n-1}\binom{m-1}{i}$. The set of $(\mathbf{b}^1, \ldots, \mathbf{b}^m) \in \mathbb{R}^n \times \ldots \times \mathbb{R}^n$ such that this *does not* hold at equality is a 0 measure set.



(A) $H_{\frac{1}{2},\frac{1}{2}} \circ N_{\ell_1}$ (B) $H_{\frac{1}{2},\frac{1}{2}} \circ \gamma_{Z(\mathbf{b}^1,\mathbf{b}^2,\mathbf{b}^3,\mathbf{b}^4)}$ (C) $H_{\frac{1}{2},\frac{1}{2},\frac{1}{2}} \circ \gamma_{Z(\mathbf{b}^1,\mathbf{b}^2,\mathbf{b}^3,\mathbf{b}^4)}$

FIGURE 2.2: We fix the $\mathbf{a}$ vectors for a two hidden layer $\mathbb{R} \to \mathbb{R}$ hard function as $\mathbf{a}^1 = \mathbf{a}^2 = (\frac{1}{2}) \in \Delta_1^1$ Left: A specific hard function induced by ℓ_1 norm: $\text{ZONOTOPE}^2_{2,2,2}[\mathbf{a}^1, \mathbf{a}^2, \mathbf{b}^1, \mathbf{b}^2]$ where $\mathbf{b}^1 = (0,1)$ and $\mathbf{b}^2 = (1,0)$. Note that in this case the function can be seen as a composition of $H_{\mathbf{a}^1,\mathbf{a}^2}$ with ℓ_1-norm $N_{\ell_1}(x) := \|x\|_1 = \gamma_{Z((0,1),(1,0))}$. Middle: A typical hard function $\text{ZONOTOPE}^2_{2,2,4}[\mathbf{a}^1, \mathbf{a}^2, \mathbf{c}^1, \mathbf{c}^2, \mathbf{c}^3, \mathbf{c}^4]$ with generators $\mathbf{c}^1 = (\frac{1}{4}, \frac{1}{2}), \mathbf{c}^2 = (-\frac{1}{2}, 0), \mathbf{c}^3 = (0, -\frac{1}{4})$ and $\mathbf{c}^4 = (-\frac{1}{4}, -\frac{1}{4})$. Note how increasing the number of zonotope generators makes the function more complex. Right: A *harder* function from $\text{ZONOTOPE}^2_{3,2,4}$ family with the same set of generators $\mathbf{c}_1, \mathbf{c}_2, \mathbf{c}_3, c_4$ but one more hidden layer ($k = 3$). Note how increasing the depth make the function more complex. (For illustrative purposes we plot only the part of the function which lies above zero.)

2. $\gamma_{Z(\mathbf{b}^1,\ldots,\mathbf{b}^m)}(\mathbf{r}) = \max_{\mathbf{x} \in Z(\mathbf{b}^1,\ldots,\mathbf{b}^m)} \langle \mathbf{r}, \mathbf{x} \rangle = \max_{\mathbf{x} \in \text{vert}(Z(\mathbf{b}^1,\ldots,\mathbf{b}^m))} \langle \mathbf{r}, \mathbf{x} \rangle$, and $\gamma_{Z(\mathbf{b}^1,\ldots,\mathbf{b}^m)}$ is therefore a piecewise linear function with $|\text{vert}(Z(\mathbf{b}^1, \ldots, \mathbf{b}^m))|$ pieces.

3. $\gamma_{Z(\mathbf{b}^1,\ldots,\mathbf{b}^m)}(\mathbf{r}) = |\langle \mathbf{r}, \mathbf{b}^1 \rangle| + \ldots + |\langle \mathbf{r}, \mathbf{b}^m \rangle|$.

Definition 8 (extremal zonotope set)**.** The set $S(n,m)$ will denote the set of $(\mathbf{b}^1, \ldots, \mathbf{b}^m) \in \mathbb{R}^n \times \ldots \times \mathbb{R}^n$ such that $|\text{vert}(Z(\mathbf{b}^1, \ldots, \mathbf{b}^m))| = \sum_{i=0}^{n-1} \binom{m-1}{i}$. $S(n,m)$ is the so-called "extremal zonotope set", which is a subset of $\mathbb{R}^{nm}$, whose complement has zero Lebesgue measure in $\mathbb{R}^{nm}$.

Lemma 2.3.8. Given any $\mathbf{b}^1, \ldots, \mathbf{b}^m \in \mathbb{R}^n$, there exists a 2-layer ReLU DNN with size $2m$ which represents the function $\gamma_{Z(\mathbf{b}^1,\ldots,\mathbf{b}^m)}(\mathbf{r})$.

Proof of Lemma 2.3.8. By Theorem 2.3.7(part 3.), $\gamma_{Z(\mathbf{b}^1,\ldots,\mathbf{b}^m)}(\mathbf{r}) = |\langle \mathbf{r}, \mathbf{b}^1 \rangle| + \ldots + |\langle \mathbf{r}, \mathbf{b}^m \rangle|$. It suffices to observe

$$|\langle \mathbf{r}, \mathbf{b}^1 \rangle| + \ldots + |\langle \mathbf{r}, \mathbf{b}^m \rangle| = \max\{\langle \mathbf{r}, \mathbf{b}^1 \rangle, -\langle \mathbf{r}, \mathbf{b}^1 \rangle\} + \ldots + \max\{\langle \mathbf{r}, \mathbf{b}^m \rangle, -\langle \mathbf{r}, \mathbf{b}^m \rangle\}.$$

□

Proposition 2.3.9. Given any tuple $(\mathbf{b}^1, \ldots, \mathbf{b}^m) \in S(n, m)$ and any point

$$(\mathbf{a}^1, \ldots, \mathbf{a}^k) \in \bigcup_{M>0} \underbrace{(\Delta_M^{w-1} \times \Delta_M^{w-1} \times \ldots \times \Delta_M^{w-1})}_{k \text{ times}},$$

the function $\text{ZONOTOPE}^n_{k,w,m}[\mathbf{a}^1, \ldots, \mathbf{a}^k, \mathbf{b}^1, \ldots, \mathbf{b}^m] := H_{\mathbf{a}^1,\ldots,\mathbf{a}^k} \circ \gamma_{Z(\mathbf{b}^1,\ldots,\mathbf{b}^m)}$ has $(m-1)^{n-1} w^k$ pieces and it can be represented by a $k+2$ layer ReLU DNN with size $2m + wk$.

Proof of Proposition 2.3.9. The fact that $\text{ZONOTOPE}^n_{k,w,m}[\mathbf{a}^1, \ldots, \mathbf{a}^k, \mathbf{b}^1, \ldots, \mathbf{b}^m]$ can be represented by a $k+2$ layer ReLU DNN with size $2m + wk$ follows from Lemmas 2.3.8 and 2.A.2. The number of pieces follows from the fact that $\gamma_{Z(\mathbf{b}^1,\ldots,\mathbf{b}^m)}$ has $\sum_{i=0}^{n-1} \binom{m-1}{i}$ distinct linear pieces by parts 1. and 2. of Theorem 2.3.7, and $H_{\mathbf{a}^1,\ldots,\mathbf{a}^k}$ has w^k pieces by Lemma 2.3.5. □

Finally, we are ready to state the main result of this section.

Theorem 2.3.10. For every tuple of natural numbers $n, k, m \geq 1$ and $w \geq 2$, there exists a family of $\mathbb{R}^n \to \mathbb{R}$ functions, which we call $\text{ZONOTOPE}^n_{k,w,m}$ with the following properties:

(i) Every $f \in \text{ZONOTOPE}^n_{k,w,m}$ is representable by a ReLU DNN of depth $k+2$ and size $2m + wk$, and has $\left(\sum_{i=0}^{n-1} \binom{m-1}{i}\right) w^k$ pieces.

(ii) Consider any $f \in \text{ZONOTOPE}^n_{k,w,m}$. If f is represented by a $(k'+1)$-layer DNN for any $k' \leq k$, then this $(k'+1)$-layer DNN has size at least $\max\left\{\frac{1}{2}(k' w^{\frac{k}{k'n}}) \cdot (m-1)^{(1-\frac{1}{n})\frac{1}{k'}} - 1 \;,\; \frac{w^{\frac{k}{k'}}}{n^{1/k'}} k'\right\}$.

(iii) The family $\text{ZONOTOPE}^n_{k,w,m}$ is in one-to-one correspondence with

$$S(n, m) \times \bigcup_{M>0} \underbrace{(\Delta_M^{w-1} \times \Delta_M^{w-1} \times \ldots \times \Delta_M^{w-1})}_{k \text{ times}}.$$

Proof of Theorem 2.3.10. Follows from Proposition 2.3.9 (and invoking Lemma 2.A.7 to get the size lowerbounds). □

Comparison to the results in (Montufar et al., 2014)

Firstly we note that the construction in Montufar et al., 2014 requires all the hidden layers to have width at least as big as the input dimensionality n. In contrast, we do not impose such restrictions and the network size in our construction is independent of the input dimensionality. Thus our result probes networks with bottleneck architectures whose complexity cant be seen from their result.

Secondly, in terms of our complexity measure, there seem to be regimes where our bound does better. One such regime, for example, is when $n \leq w < 2n$ and $k \in \Omega(\frac{n}{\log(n)})$, by setting in our construction $m < n$.

Thirdly, it is not clear to us whether the construction in Montufar et al., 2014 gives a smoothly parameterized family of functions other than by introducing small perturbations of the construction in their paper. In contrast, we have a smoothly parameterized family which is in one-to-one correspondence with a well-understood manifold like the higher-dimensional torus.

2.4 Training 2-layer $\mathbb{R}^n \to \mathbb{R}$ ReLU DNNs to global optimality

In this section we consider the following empirical risk minimization problem. Given D data points $(x_i, y_i) \in \mathbb{R}^n \times \mathbb{R}$, $i = 1, \ldots, D$, find the function f represented by 2-layer $\mathbb{R}^n \to \mathbb{R}$ ReLU DNNs of width w, that minimizes the following optimization problem,

$$\min_{f \in \mathscr{F}_{\{n,w,1\}}} \frac{1}{D} \sum_{i=1}^{D} \ell(f(x_i), y_i) \quad \equiv \min_{T_1 \in \mathscr{A}_n^w,\, T_2 \in \mathscr{L}_w^1} \frac{1}{D} \sum_{i=1}^{D} \ell\big(\, T_2(\sigma(T_1(x_i))), y_i \,\big) \tag{2.3}$$

where $\ell : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$ is a convex *loss function* (common loss functions are the squared loss, $\ell(y, y') = (y - y')^2$, and the hinge loss function given by $\ell(y, y') = \max\{0, 1 - yy'\}$). Our main result of this section gives an algorithm to solve the above empirical risk minimization problem to global optimality.

Theorem 2.4.1. There exists an algorithm to find a global optimum of Problem 2.3 in time $O(2^w(D)^{nw}\text{poly}(D, n, w))$. Note that the running time $O(2^w(D)^{nw}\text{poly}(D, n, w))$ is polynomial in the data size D for fixed n, w.

Proof Sketch: Before giving the full proof of Theorem 2.4.1 below here we first provide a sketch of it. When the empirical risk minimization problem is viewed as an optimization problem in the space of weights of the ReLU DNN, it is a nonconvex, quadratic problem. However, one can instead search over the space of functions representable by 2-layer DNNs by writing them in the form similar to (2.1). This breaks the problem into two parts: a combinatorial search and then a convex problem that is essentially linear regression with linear inequality constraints. This enables us to guarantee global optimality.

Let $T_1(x) = Ax + b$ and $T_2(y) = a' \cdot y$ for $A \in \mathbb{R}^{w \times n}$ and $b, a' \in \mathbb{R}^w$. If we denote the i-th row of the matrix A by a^i, and write b_i, a'_i to denote the i-th coordinates of the vectors b, a' respectively, due to

Algorithm 1 Empirical Risk Minimization

```
function ERM(𝒟)                                   ▷ Where 𝒟 = {(x_i, y_i)}_{i=1}^D ⊂ ℝ^n × ℝ
    𝒮 = {+1, −1}^w                                ▷ All possible instantiations of top layer weights
    𝒫^i = {(P_+^i, P_−^i)},  i = 1, ..., w        ▷ All possible partitions of data into two parts
    𝒫 = 𝒫^1 × 𝒫^2 × ··· × 𝒫^w
    count = 1                                     ▷ Counter
    for s ∈ 𝒮 do
        for {(P_+^i, P_−^i)}_{i=1}^w ∈ 𝒫 do
            loss(count) = { min_{ã,b̃} Σ_{j=1}^D Σ_{i:j∈P_+^i} ℓ(y_j, s_i(ã^i · x_j + b̃_i))
                          { ã^i · x_j + b̃_i ≤ 0   ∀j ∈ P_−^i
                          { ã^i · x_j + b̃_i ≥ 0   ∀j ∈ P_+^i
            count++
        end for
        OPT = argmin loss(count)
    end for
    return {ã}, {b̃}, s corresponding to OPT's iterate
end function
```

homogeneity of ReLU gates, the network output can be represented as

$$f(x) = \sum_{i=1}^{w} a'_i \max\{0, a^i \cdot x + b_i\} = \sum_{i=1}^{w} s_i \max\{0, \tilde{a}^i \cdot x + \tilde{b}_i\}.$$

where $\tilde{a}^i \in \mathbb{R}^n$, $\tilde{b}_i \in \mathbb{R}$ and $s_i \in \{-1, +1\}$ for all $i = 1, \ldots, w$.

For any hidden node $i \in \{1 \ldots, w\}$, the pair $(\tilde{a}^i, \tilde{b}_i)$ induces a partition $\mathscr{P}^i := (P_+^i, P_-^i)$ on the dataset, given by $P_-^i = \{j : \tilde{a}^i \cdot x_j + \tilde{b}_i \leq 0\}$ and $P_+^i = \{1, \ldots, D\} \backslash P_-^i$. Algorithm 1 proceeds by generating all combinations of the partitions $\mathscr{P}^i$ as well as the top layer weights $s \in \{+1, -1\}^w$, and minimizing the loss $\sum_{j=1}^{D} \sum_{i:j \in P_+^i} \ell(s_i(\tilde{a}^i \cdot x_j + \tilde{b}_i), y_j)$ subject to the constraints $\tilde{a}^i \cdot x_j + \tilde{b}_i \leq 0 \quad \forall j \in P_-^i$ and $\tilde{a}^i \cdot x_j + \tilde{b}_i \geq 0 \quad \forall j \in P_+^i$ which are imposed for all $i = 1, \ldots, w$,which is a convex program.

Proof of Theorem 2.4.1. Let $\ell : \mathbb{R} \to \mathbb{R}$ be any convex loss function, and let $(x_1, y_1), \ldots, (x_D, y_D) \in \mathbb{R}^n \times \mathbb{R}$ be the given D data points. As stated in (2.3), the problem requires us to find an affine transformation $T_1 : \mathbb{R}^n \to \mathbb{R}^w$ and a linear transformation $T_2 : \mathbb{R}^w \to \mathbb{R}$, so as to minimize the empirical loss as stated in (2.3). Note that T_1 is given by a matrix $A \in \mathbb{R}^{w \times n}$ and a vector $b \in \mathbb{R}^w$ so that $T(x) = Ax + b$ for all $x \in \mathbb{R}^n$. Similarly, T_2 can be represented by a vector $a' \in \mathbb{R}^w$ such that $T_2(y) = a' \cdot y$ for all $y \in \mathbb{R}^w$. If we denote the i-th row of the matrix A by a^i, and write b_i, a'_i to denote the i-th coordinates of the vectors b, a' respectively, we can write the function represented by this network as

$$f(x) = \sum_{i=1}^{w} a'_i \max\{0, a^i \cdot x + b_i\} = \sum_{i=1}^{w} \operatorname{sgn}(a'_i) \max\{0, (|a'_i| a^i) \cdot x + |a'_i| b_i\}.$$

In other words, the family of functions over which we are searching is of the form

$$f(x) = \sum_{i=1}^{w} s_i \max\{0, \tilde{a}^i \cdot x + \tilde{b}_i\} \tag{2.4}$$

where $\tilde{a}^i \in \mathbb{R}^n$, $b_i \in \mathbb{R}$ and $s_i \in \{-1, +1\}$ for all $i = 1, \ldots, w$.

We now make the following observation. For a given data point (x_j, y_j) if $\tilde{a}^i \cdot x_j + \tilde{b}_i \leq 0$, then the i-th term of (2.4) does not contribute to the loss function for this data point (x_j, y_j). Thus, for every data point (x_j, y_j), there exists a set $S_j \subseteq \{1, \ldots, w\}$ such that $f(x_j) = \sum_{i \in S_j} s_i(\tilde{a}^i \cdot x_j + \tilde{b}_i)$. In particular, if we are given the set S_j for (x_j, y_j), then the expression on the right hand side of (2.4) reduces to a linear function of $\tilde{a}^i, \tilde{b}_i$. For any fixed $i \in \{1, \ldots, w\}$, these sets S_j induce a partition of the data set into two parts. In particular, we define $P^i_+ := \{j : i \in S_j\}$ and $P^i_- := \{1, \ldots, D\} \setminus P^i_+$. Observe now that this partition is also induced by the hyperplane given by $\tilde{a}^i, \tilde{b}_i$: $P^i_+ = \{j : \tilde{a}^i \cdot x_j + \tilde{b}_i > 0\}$ and $P^i_+ = \{j : \tilde{a}^i \cdot x_j + \tilde{b}_i \leq 0\}$. Our strategy will be to *guess* the partitions P^i_+, P^i_- for each $i = 1, \ldots, w$, and then do linear regression with the constraint that regression's decision variables $\tilde{a}^i, \tilde{b}_i$ induce the guessed partition.

More formally, the algorithm does the following. For each $i = 1, \ldots, w$, the algorithm guesses a partition of the data set (x_j, y_j), $j = 1, \ldots, D$ by a hyperplane. Let us label the partitions as follows (P^i_+, P^i_-), $i = 1, \ldots, w$. So, for each $i = 1, \ldots, w$, $P^i_+ \cup P^i_- = \{1, \ldots, D\}$, P^i_+ and P^i_- are disjoint, and there exists a vector $c \in \mathbb{R}^n$ and a real number δ such that $P^i_- = \{j : c \cdot x_j + \delta \leq 0\}$ and $P^i_+ = \{j : c \cdot x_j + \delta > 0\}$. Further, for each $i = 1, \ldots, w$ the algorithm selects a vector s in $\{+1, -1\}^w$.

For a fixed selection of partitions (P^i_+, P^i_-), $i = 1, \ldots, w$ and a vector s in $\{+1, -1\}^w$, the algorithm solves the following convex optimization problem with decision variables $\tilde{a}^i \in \mathbb{R}^n$, $\tilde{b}_i \in \mathbb{R}$ for $i = 1, \ldots, w$ (thus, we have a total of $(n+1) \cdot w$ decision variables). The feasible region of the optimization is given by the constraints

$$\begin{aligned} \tilde{a}^i \cdot x_j + \tilde{b}_i &\leq 0 \quad \forall j \in P^i_- \\ \tilde{a}^i \cdot x_j + \tilde{b}_i &\geq 0 \quad \forall j \in P^i_+ \end{aligned} \tag{2.5}$$

which are imposed for all $i = 1, \ldots, w$. Thus, we have a total of $D \cdot w$ constraints. Subject to these constraints we minimize the objective $\sum_{j=1}^{D} \sum_{i : j \in P^i_+} \ell(s_i(\tilde{a}^i \cdot x_j + \tilde{b}_i), y_j)$. Assuming the loss function ℓ is a convex function in the first argument, the above objective is a convex function. Thus, we have to minize a convex objective subject to the linear inequality constraints from (2.5).

We finally have to count how many possible partitions (P^i_+, P^i_-) and vectors s the algorithm has to search through. It is well-known (Matousek, 2002) that the total number of possible hyperplane partitions of a set of size D in $\mathbb{R}^n$ is at most $2\binom{D}{n} \leq D^n$ whenever $n \geq 2$. Thus with a guess for each

$i = 1, \ldots, w$, we have a total of at most D^{nw} partitions. There are 2^w vectors s in $\{-1, +1\}^w$. This gives us a total of $2^w D^{nw}$ guesses for the partitions (P_+^i, P_-^i) and vectors s. For each such guess, we have a convex optimization problem with $(n+1) \cdot w$ decision variables and $D \cdot w$ constraints, which can be solved in time $\text{poly}(D, n, w)$. Putting everything together, we have the running time claimed in the statement.

The above argument holds only for $n \geq 2$, since we used the inequality $2\binom{D}{n} \leq D^n$ which only holds for $n \geq 2$. For $n = 1$, a similar algorithm can be designed, but one which uses the characterization achieved in Theorem 2.2.2.

Let $\ell : \mathbb{R} \to \mathbb{R}$ be any convex loss function, and let $(x_1, y_1), \ldots, (x_D, y_D) \in \mathbb{R}^2$ be the given D data points. Using Theorem 2.2.2, to solve problem (2.3) it suffices to find a $\mathbb{R} \to \mathbb{R}$ piecewise linear function f with w pieces that minimizes the total loss. In other words, the optimization problem (2.3) is equivalent to the problem

$$\min \left\{ \sum_{i=1}^{D} \ell(f(x_i), y_i) : f \text{ is piecewise linear with } w \text{ pieces} \right\}. \tag{2.6}$$

We now use the observation that fitting piecewise linear functions to minimize loss is just a step away from linear regression, which is a special case where the function is contrained to have exactly one affine linear piece. Our algorithm will first guess the optimal partition of the data points such that all points in the same class of the partition correspond to the same affine piece of f, and then do linear regression in each class of the partition. Alternatively, one can think of this as guessing the interval (x_i, x_{i+1}) of data points where the $w-1$ breakpoints of the piecewise linear function will lie, and then doing linear regression between the breakpoints.

More formally, we parametrize piecewise linear functions with w pieces by the w slope-intercept values $(a_1, b_1), \ldots, (a_2, b_2), \ldots, (a_w, b_w)$ of the w different pieces. This means that between breakpoints j and $j+1$, $1 \leq j \leq w-2$, the function is given by $f(x) = a_{j+1}x + b_{j+1}$, and the first and last pieces are $a_1 x + b_1$ and $a_w x + b_w$, respectively.

Define $\mathcal{I}$ to be the set of all $(w-1)$-tuples $(i_1, \ldots, i_{w-1})$ of natural numbers such that $1 \leq i_1 \leq \ldots \leq i_{w-1} \leq D$. Given a fixed tuple $I = (i_1, \ldots, i_{w-1}) \in \mathcal{I}$, we wish to search through all piecewise linear functions whose breakpoints, in order, appear in the intervals $(x_{i_1}, x_{i_1+1}), (x_{i_2}, x_{i_2+1})$, $\ldots, (x_{i_{w-1}}, x_{i_{w-1}+1})$. Define also $\mathcal{S} = \{-1, 1\}^{w-1}$. Any $S \in \mathcal{S}$ will have the following interpretation: if $S_j = 1$ then $a_j \leq a_{j+1}$, and if $S_j = -1$ then $a_j \geq a_{j+1}$. Now for every $I \in \mathcal{I}$ and $S \in \mathcal{S}$, requiring a

piecewise linear function that respects the conditions imposed by I and S is easily seen to be equivalent to imposing the following linear inequalities on the parameters $(a_1, b_1), \ldots, (a_2, b_2), \ldots, (a_w, b_w)$:

$$\begin{aligned} S_j(b_{j+1} - b_j - (a_j - a_{j+1})x_{i_j}) &\geq 0 \\ S_j(b_{j+1} - b_j - (a_j - a_{j+1})x_{i_j+1}) &\leq 0 \\ S_j(a_{j+1} - a_j) &\geq 0 \end{aligned} \tag{2.7}$$

Let the set of piecewise linear functions whose breakpoints satisfy the above be denoted by $\text{PWL}^1_{I,S}$ for $I \in \mathcal{I}, S \in \mathcal{S}$.

Given a particular $I \in \mathcal{I}$, we define

$$\begin{aligned} D_1 &:= \{x_i : i \leq i_1\}, \\ D_j &:= \{x_i : i_{j-1} < i \leq i_1\} \quad j = 2, \ldots, w-1, \cdot \\ D_w &:= \{x_i : i > i_{w-1}\} \end{aligned}$$

Observe that

$$\min\{\sum_{i=1}^{D} \ell(f(x_i) - y_i) : f \in \text{PWL}^1_{I,S}\} = \min\{\sum_{j=1}^{w} \left(\sum_{i \in D_j} \ell(a_j \cdot x_i + b_j - y_i) \right) : (a_j, b_j) \text{ satisfy (2.7)}\} \tag{2.8}$$

The right hand side of the above equation is the problem of minimizing a convex objective subject to linear constraints. Now, to solve (2.6), we need to simply solve the problem (2.8) for all $I \in \mathcal{I}, S \in \mathcal{S}$ and pick the minimum. Since $|\mathcal{I}| = \binom{D}{w} = O(D^w)$ and $|\mathcal{S}| = 2^{w-1}$ we need to solve $O(2^w \cdot D^w)$ convex optimization problems, each taking time $O(\text{poly}(D))$. Therefore, the total running time is $O((2D)^w \text{poly}(D))$. □

2.4.1 Discussion on the complexity of solving ERM on deep-nets

The running time of the algorithm (Algorithm 1) that we gave above to find the exact global minima of a two layer ReLU-DNN is exponential in the input dimension n and the number of hidden nodes w. The exponential dependence on n can not be removed unless $P = NP$; see Shalev-Shwartz and Ben-David, 2014; Blum and Rivest, 1992; DasGupta, Siegelmann, and Sontag, 1995; Dey, Wang, and Xie, 2018. However, we are not aware of any complexity results which would rule out the possibility of an algorithm which trains to global optimality in time that is polynomial in the data size and/or the number of hidden nodes, assuming that the input dimension is a fixed constant. Resolving this dependence on network size would be another step towards clarifying the theoretical complexity of training ReLU DNNs and is a good open question for future research, in our opinion. Thus our

training result of solving the ERM on depth 2 nets in time polynomial in the number of data points is a step towards resolving this gap in the complexity literature.

A related result for *improperly* learning ReLUs has been recently obtained in Goel et al., 2016. In contrast, our algorithm returns a ReLU DNN from the class being learned. Another difference is that their result considers the notion of *reliable learning* as opposed to the empirical risk minimization objective considered in (2.3) for which we give a quick definition below,

Definition 9. Suppose distribution $\mathcal{D}$ is supported on $X \times [0,1]$. For $[0,1] \subseteq Y'$ let $h : X \to Y'$ be some function and let $\ell : Y' \times [0,1] \to \mathbb{R}^+$ be a loss function. The we define two notions of expected loss,

$$L_{=0}(h, \mathcal{D}) = \mathbb{P}_{(\mathbf{x},y)\sim\mathcal{D}}[h(\mathbf{x}) \neq 0 \text{ and } y = 0]$$
$$L_{>0}(h, \mathcal{D}) = \mathbb{E}_{(\mathbf{x},y)\sim\mathcal{D}}[\ell(h(\mathbf{x}), y).1_{y>0}]$$

We say that a concept class $\mathscr{C} \subseteq [0,1]^X$ is "reliably agnostically learnable with respect to a loss function, $\ell : Y' \times [0,1] \to \mathbb{R}^+$" (where $[0,1] \subseteq Y'$) if for every $\epsilon, \delta > 0$ there exists a learning algorithm which satisfies the following :

That $\forall$ distributions $\mathcal{D}$ over $X \times [0,1]$ given access to examples drawn from $\mathcal{D}$, the algorithm outputs a hypothesis $h : X \to Y$ such that,

$$L_{=0}(h, \mathcal{D}) \leq \epsilon \text{ and } L_{>0}(h, \mathcal{D}) \leq \epsilon + \min_{c\in\mathscr{C}'(\mathcal{D})} L_{>0}(c)$$

where,

$$\mathscr{C}'(\mathcal{D}) = \{c \in \mathscr{C} | L_{=0}(c, \mathcal{D}) = 0\}$$

Further if if $X \subseteq \mathbb{R}^n$ and s is a parameter that captures the representation complexity (i.e description length) of concepts in $c \in \mathscr{C}$ then we say that $\mathscr{C}$ is "efficiently reliably agnostically learnable to error ϵ" if the running time of the above algorithm that is supposed to exist is $poly(n, s, \frac{1}{\delta})$.

Asking or $L_{=0}(h, \mathcal{D})$ to be low captures mathematically the idea of trying to minimize the rate of "false positives".

Perhaps a big breakthrough would be to get optimal training algorithms for DNNs with two or more hidden layers and this seems like a substantially harder nut to crack. We end this discussion by

pointing out some recent progress towards that which has been made in Boob, Dey, and Lan, 2018.

2.5 Understanding neural functions over Boolean inputs

The classic paper Maass, 1997, established complexity results for the entire class of functions represented by circuits where the gates can come from a very general family while the inputs are restricted to discrete domains. This is complemented by papers that study a very specific family of gates such as the sigmoid gate or the LTF gate ($\mathbb{R} \ni y \mapsto \mathbf{1}_{y \geq 0}$) (Impagliazzo, Paturi, and Saks, 1997), (Siu, Roychowdhury, and Kailath, 1994; Sherstov, 2007; Krause and Pudlák, 1994), (Buhrman, Vereshchagin, and Wolf, 2007; Sherstov, 2009; Razborov and Sherstov, 2010; Bun and Thaler, 2016). Many associated results can also be found in these reviews like Lee, Shraibman, et al., 2009 and Razborov, 1992. Recent circuit complexity results in Kane and Williams, 2016, Tamaki, 2016, Chen, Santhanam, and Srinivasan, 2016, Kabanets, Kane, and Lu, 2017 stand out as significant improvements over known lower (and upper) bounds on circuit complexity with threshold gates. The results of Maass, 1997 also show that very general families of neural networks can be converted into circuits with only LTF gates with at most a constant factor blow up in depth and polynomial blow up in size of the circuits.

Some of the prior results which apply to general gates, such as the ones in Maass, 1997, also apply to ReLU gates, because those results apply to gates that compute a piecewise polynomial function (ReLU is a piecewise linear function with only two pieces). However, as witnessed by results on LTF gates, one can usually make much stronger claims about specific classes of gates. The main focus of this work is to study circuits computing Boolean functions mapping $\{-1,1\}^m \to \{-1,1\}$ which use ReLU gates in their intermediate layers, and have an LTF gate at the output node (to ensure that the output is in $\{-1,1\}$). We remark that using an LTF gate at the output node while allowing more general analog gates in the intermediate nodes is a standard practice when studying the Boolean complexity of analog gates (see, for example, Maass, 1997).

Other than Williams, 2018, we are not aware of an analysis of lower bounds for ReLU circuits when applied to only Boolean inputs. In contrast, there has been recent work on the analysis of such circuits when viewed as a function from $\mathbb{R}^n$ to $\mathbb{R}$ (i.e., allowing real inputs and output). From Eldan and Shamir, 2016 and Daniely, 2017 (with restrictions on the domain and the weights) we know of (super-)exponential lowerbounds on the size of Sum-of-ReLU circuits for certain easy Sum-of-ReLU-of-ReLU functions . Depth v/s size tradeoffs for such circuits have recently also been studied in Telgarsky, 2016a; Hanin, 2017; Liang and Srikant, 2016; Yarotsky, 2016; Safran and Shamir, 2016 and in this chapter so far. But to the best of our knowledge no lowerbounds scaling exponentially with

the dimension are known for analog deep neural networks of depths more than 2.

In what follows, the *depth* of a circuit will be the length of the longest path from the output node to an input variable, and the *size* of a circuit will be the total number of gates in the circuit. We will also use the notation *Sum-of-ReLU* to refer to circuits whose inputs feed into a single layer of ReLU gates, whose outputs are combined into a weighted sum to give the final output. Similarly, Sum-of-ReLU-of-ReLU denotes the circuit with depth 3, where the output node is a simple weighted sum, and the intermediate gates are all ReLU gates in the two "hidden" layers. We analogously define Sum-of-LTF, LTF-of-LTF, LTF-of-ReLU, LTF-of-LTF-of-LTF, LTF-of-ReLU-of-ReLU and so on. We will also use the notation LTF-of-(ReLU)^k for a circuit of the form LTF-of-ReLU-of-RELU-...-ReLU with $k \geq 1$ levels of ReLU gates.

2.5.1 Statement and discussion of our results over Boolean inputs

Boolean v/s real inputs. We begin our study with the following observation which shows that ReLU circuits have markedly different behaviour when the inputs are restricted to be Boolean, as opposed to arbitrary real inputs. Since AND and OR gates can both be implemented by ReLU gates, it follows that *any* Boolean function can be implemented by a ReLU-of-ReLU circuit. In fact, it is not hard to show something slightly stronger:

Lemma 2.5.1. Any function $f : \{-1,1\}^n \to \mathbb{R}$ can be implemented by a Sum-of-ReLU circuit using at most $\min\{2^n, \sum_{\hat{f}(S)\neq 0} |S|\}$ number of ReLU gates, where $\hat{f}(S)$ denotes the Fourier coefficient of f for the set $S \subseteq \{1,\ldots,n\}$.

The Lemma follows by observing that the indicator functions of each vertex of the Boolean hypercube $\{-1,1\}^n$ can be implemented by a single ReLU gate, and the parity function on k variables can be implemented by k ReLU gates (see Appendix 2.D). Thus, if one does not restrict the size of the circuit, then Sum-of-ReLU circuits can represent any pseudo-Boolean function. In contrast, we will now show that if one allows real inputs, then there exist functions with just 2 inputs (i.e., $n = 2$) which cannot be represented by any Sum-of-ReLU circuit, no matter how large.

Proposition 2.5.2. The function $\max\{0, x_1, x_2\}$ cannot be computed by any Sum-of-ReLU circuit, no matter how many ReLU gates are used. It can be computed by a Sum-of-ReLU-of-ReLU circuit.

The first part of the above proposition (the impossibility result) is proved in Appendix 2.B. The second part follows from Lemma 2.2.1, which stated that any $\mathbb{R}^n \to \mathbb{R}$ function that can be implemented by

a circuit of ReLU gates, can always be implemented with at most $\lceil \log(n+1) \rceil$ layers of ReLU gates (with a weighted Sum to give the final output).

Restricting to Boolean inputs. From this point on, we will focus entirely on the situation where the inputs to the circuits are restricted to $\{-1, 1\}$. One motivation behind our results is the desire to understand the strength of the ReLU gates vis-a-vis LTF gates. It is not hard to see that any circuit with LTF gates can be simulated by a circuit with ReLU gates with at most a constant blow-up in size (because a single LTF gate can be simulated by 2 ReLU gates when the inputs are a discrete set – see Appendix 2.C). The question is whether ReLU gates can do significantly better than LTF gates in terms of depth and/or size.

A quick observation is that Sum-of-ReLU circuits can be linearly (in the dimension n) smaller than Sum-of-LTF circuits. More precisely,

Proposition 2.5.3. The function $f : \{-1, 1\}^n \to \mathbb{R}$ given by $f(x) = \sum_{i=1}^{n} 2^i \left(\frac{1+x_i}{2}\right)$ can be implemented by a Sum-of-ReLU circuit with 2 ReLU gates, and any Sum-of-LTF that implements f needs $\Omega(n)$ gates.

The above result follows from the following two facts: 1) any linear function is implementable by 2 ReLU gates, and 2) any Sum-of-LTF circuit with w LTF gates gives a piecewise constant function that takes at most 2^w different values. Since f takes 2^n different values (it evaluates every vertex of the Boolean hypercube to the corresponding natural number expressed in binary), we need $w \geq n$ gates.

In the context of these preliminary results, we now state our main contributions. For the next result we recall the definition of the Andreev function (Andreev, 1987) which has previously many times been used to prove computational lower bounds (Paterson and Zwick, 1993; Impagliazzo and Naor, 1988; Impagliazzo, Meka, and Zuckerman, 2012).

Definition 10 (Andreev's function). The Andreev's function is the following mapping,

$$A_n : \{0,1\}^{\lfloor \frac{n}{2} \rfloor} \times \{0,1\}^{\lfloor \log(\frac{n}{2}) \rfloor \times \lfloor \frac{n}{2\lfloor \log(\frac{n}{2}) \rfloor} \rfloor} \longrightarrow \{0,1\}$$
$$(\mathbf{x}, [a_{ij}]) \longmapsto x_{\mathrm{bin}(z([a_{ij}]))}$$

where $z([a_{ij}]) = \{(\sum_{j=1}^{\lfloor \frac{n}{2\lfloor \log(\frac{n}{2}) \rfloor} \rfloor} a_{ij}) \mod 2\}_{i=1,2,..,\lfloor \log(\frac{n}{2}) \rfloor}$ is the binary string constructed by noting down the odd/even parity of each of the row sums in the matrix $[a_{ij}]$ and "bin" is the function that gives the decimal number that can be represented by its input bit string.

Kane and Williams, 2016 have recently established the first super linear lower bounds for approximating the Andreev function using LTF-of-LTF circuits. In the following theorem we show that their techniques can be adapted to also establish an almost linear lower bound on the size of LTF-of-ReLU circuits approximating this Andreev function with no restriction on the weights $\mathbf{w}, b$ for each gate.

Theorem 2.5.4. For any $\delta \in (0, \frac{1}{2})$, there exists $N(\delta) \in \mathbb{N}$ such that for all $n \geq N(\delta)$ and $\epsilon > \sqrt{\frac{2\log^{\frac{2}{2-\delta}}(n)}{n}}$, any LFT-of-ReLU circuit on n bits that matches the Andreev function on $n-$bits for at least $1/2 + \epsilon$ fraction of the inputs, has size $\Omega(\epsilon^{2(1-\delta)} n^{1-\delta})$.

It is well known that proving lower bounds without restrictions on the weights is much more challenging even in the context of LTF circuits. In fact, the recent results in Kane and Williams, 2016 are the first superlinear lower bounds for LTF circuits with no restrictions on the weights. With restrictions on some or all the weights, e.g., assuming poly(n) bounds on the weights (typically termed the "small weight asssumption") in certain layers, exponential lower bounds have been established for LTF circuits (Hajnal et al., 1987; Impagliazzo, Paturi, and Saks, 1997; Sherstov, 2009; Sherstov, 2011). Our next results are of this flavor: under certain kinds of weight restrictions, we prove exponential size lower bounds on the size of LTF-of-(ReLU)$^{d-1}$ circuits. *We emphasize that our weight restrictions are assumed only on the bottom layer (closest to the input). The other layers can have gates with unbounded weights.* Nevertheless, our weight restrictions are somewhat unconventional.

Definition 11. (**The polyhedral cones** $P_{m,\sigma}$) Let $m \in \mathbb{N}$ and σ be any permutation of $\{1, \ldots, 2^m\}$. Let us also consider an arbitrary sequencing $\{\mathbf{x}^1, \ldots, \mathbf{x}^{2^m}\}$ of the vertices of the hypercube $\{-1, 1\}^m$. Define the following polyhedral cone,

$$P_{m,\sigma} := \{\mathbf{a} \in \mathbb{R}^m : \langle \mathbf{a}, \mathbf{x}^{\sigma(1)} \rangle \leq \langle \mathbf{a}, \mathbf{x}^{\sigma(2)} \rangle \leq \ldots \langle \mathbf{a}, \mathbf{x}^{\sigma(2^m)} \rangle\}.$$

In words, $P_{m,\sigma}$ is the set of all linear objectives that order the vertices of the m-dimensional hypercube in the order specified by σ. □

Definition 12. (**Our weight restriction condition**) Below, we shall be considering circuits on $2m$ inputs which come partitioned into two blocks $(\mathbf{x}, \mathbf{y})$ so that $\mathbf{x}, \mathbf{y} \in \{-1, 1\}^m$. The weight restriction

we impose is that there exist permutations σ_1 and σ_2 of $\{1,\ldots,2^m\}$ such that for *each ReLU gate in the bottom layer* mapping as, $(\mathbf{x},\mathbf{y}) \mapsto \max\{0, b + \langle \mathbf{w}_1, \mathbf{x}\rangle + \langle \mathbf{w}_2, \mathbf{y}\rangle\}$, for some bias value of b and weight vectors $\mathbf{w}_1$ and $\mathbf{w}_2$, satisfy the following two conditions, **(1)** $\mathbf{w}_i \in P_{m,\sigma_i}$ for $i = 1, 2$ (see Definition 11)and **(2)** all weights are integers with magnitude bounded by some $W > 0$.

We emphasize the existence of a single σ defining a single polyhedral cone $P_{m,\sigma}$ which contains all the weight vectors corresponding to $\mathbf{x}$ in the bottom most layer of the net (similarly for all the weights corresponding to $\mathbf{y}$). But the two cones, one for $\mathbf{x}$ and one for $\mathbf{y}$, are allowed to be different. □

Remark. One can see that $\mathbb{R}^m$ is a disjoint union of the different face-sharing polyhedral cones $P_{m,\sigma}$ obtained for different $\sigma \in S_{2m}$. Thus part (1) of the above weight restriction is equivalent to asking all the weight vectors in the bottom layer of the net corresponding to $\mathbf{x}$ part of the input to lie in any one of these special cones (and similarly for the $\mathbf{y}$ part of the input).

Let OMB is the ODD-MAX-BIT function which is a ± 1 threshold gate which evaluates to -1 on say a $n-$bit input $\mathbf{x}$ if $\sum_{i=1}^{n}(-1)^{i+1}2^i(1+x_i) \geq \frac{1}{2}$. We will prove our lower bounds against the function proposed by Arkadev Chattopadhyay and Nikhil Mande in Chattopadhyay and Mande, 2017,

$$g : \mathrm{OMB}_n^0 \circ \mathrm{OR}_{n^{\frac{1}{3}}+\log n} \circ \mathrm{XOR}_2 : \{-1,1\}^{2(n^{\frac{4}{3}}+n\log n)} \to \{-1,1\} \tag{2.9}$$

which we will refer to as the Chattopadhyay-Mande function in the remainder of the paper. Here we use the notation from Chattopadhyay and Mande, 2017 whereby if p_m and q_n are two Boolean functions taking m and n bits respectively for input, we denote a composition of them as, $p_m \circ q_n : \{-1,1\}^{mn} \to \{-1,1\}$. Here its understood that the input implicitly comes grouped into m blocks of size n on each of which q acts and p_m acts on the $m-$tuple of outputs of these q_n functions.

We show the following exponential lowerbound against this Chattopadhyay-Mande function.

Theorem 2.5.5. Let $m, d, W \in \mathbb{N}$. Any depth d LTF-of-(ReLU)$^{d-1}$ circuits on $2m$ bits such that the weights in the bottom layer are restricted as per Definition 12 that implements the Chattopadhyay-Mande function on $2m$ bits will require a circuit size of,

$$\Omega\left((d-1)\left[\frac{2^{m^{\frac{1}{8}}}}{mW}\right]^{\frac{1}{(d-1)}}\right).$$

Consequently, one obtains the same size lower bounds for circuits with only LTF gates of depth d.

Remark. Note that this is an exponential in dimension size lowerbound for even super-polynomially growing bottom layer weights (and additional constraints as per Definition 12) and upto depths scaling as $d = O(m^{\xi})$ for any $\xi < \frac{1}{8}$.

We note that the Chattopadhyay-Mande function can be represented by an $O(m)$ size LTF-of-LTF circuit with no restrictions on weights (see Theorem 2.5.6 below). In light of this fact, Theorem 2.5.5 is somewhat surprising as it shows that for the purpose of representing Boolean functions a deep ReLU circuit (ending in a LTF) gate can get exponentially weakened when just its bottom layer weights are restricted as per Definition 12, even if the integers are allowed to be super-polynomially large. Moreover, the lower bounds also hold of LTF circuits of arbitrary depth d, under the same weight restrictions on the bottom layer. We are unaware of any exponential lower bounds on LTF circuits of arbitrary depth under any kind of weight restrictions.

We will use the method of sign-rank to obtain the exponential lowerbounds in Theorems 2.5.5. The sign-rank of a real matrix A with all non-zero entries is the least rank of a matrix B of the same dimension with all non-zero entries such that for each entry (i, j), $sign(B_{ij}) = sign(A_{ij})$. For a Boolean function f mapping, $f : \{-1, 1\}^m \times \{-1, 1\}^m \to \{-1, 1\}$ one defines the "sign-rank of f" as the sign-rank of the $2^m \times 2^m$ dimensional matrix $[f(\mathbf{x}, \mathbf{y})]_{\mathbf{x},\mathbf{y}\in\{-1,1\}^m}$. This notion of a sign-rank has been used to great effect in diverse fields from communication complexity to circuit complexity to learning theory. Explicit matrices with a high sign-rank were not known till the breakthrough work by Forster, Forster, 2002. Forster et. al. showed elegant use of this complexity measure to show exponential lowerbounds against LTF-of-MAJ circuits in Forster et al., 2001. Lot of the previous literature about sign-rank has been reviewed in the book Lokam et al., 2009. Most recently Chattopadhyay and Mande, 2017 have proven a strict containment of LTF-of-MAJ in LTF-of-LTF. The following theorem statement is a combination of their Theorem 5.2 and intermediate steps in their Corollary 1.2,

Theorem 2.5.6 (Chattopadhyay-Mande (2017)).
The Chattopadhyay-Mande function g in equation 2.9 can be represented by a linear sized LTF-of-LTF circuit and sign-rank$(g) \geq 2^{n^{\frac{1}{81}}-3}$

In Appendix 2.E we will prove our Theorem 2.5.5 by showing a small upper bound on the sign-rank of LTF-of-(ReLU)$^{d-1}$ circuits which have their bottom most layer's weight restricted as given in Definition 12.

2.6 Lower bounds for LTF-of-ReLU against the Andreev function (Proof of Theorem 2.5.4)

We will use the classic "method of random restrictions" (Subbotovskaya, 1961; stad, 1998; Hastad, 1986; Yao, 1985; Rossman, 2008) to show a lowerbound for weight unrestricted LTF-of-ReLU circuits for representing the Andreev function. The basic philosophy of this method is to take any arbitrary LTF-of-ReLU circuit which supposedly matches the Andreev function on a large fraction of the inputs and to randomly fix the values on some of its input coordinates and also do the same fixing on the same coordinates of the input to the Andreev function. Then we show that upon doing this restriction the Andreev function collapses to an arbitrary Boolean function on the remaining inputs (what it collapses to depends on what values were fixed on its inputs that got restricted). But on the other hand we show that the LTF-of-ReLU collapses to a circuit which is of such a small size that with high-probability it cannot possibly approximate a randomly chosen Boolean function on the remaining inputs. This contradiction leads to a lowerbound.

There are two important concepts towards implementing the above idea. First one is about being able to precisely define as to when can a ReLU gate upon a partial restriction of its inputs be considered to be removable from the circuit. Once this notion is clarified it will automatically turn out that doing random restrictions on ReLU is the same as doing random restriction on a LTF gate as was recently done in Kane and Williams, 2016. And secondly it needs to be true that at any fixed size, LTF-of-ReLU circuits cannot represent too many of all the Boolean functions possible at the same input dimension. For this very specific case of LTF-of-ReLU circuits where ReLU gates necessarily have a fan-out of 1, Theorem 2.1 in Maass, 1997 applies and we have from there that LTF-of-ReLU circuits over $n-$bits with w ReLU gates can represent at most $N = 2^{O((wn+w+w+1+1)^2 \log(wn+w+w+1+1))} = 2^{O((wn+2w+2)^2 \log(wn+2w+2))}$ number of Boolean functions. We note that slightly departing from the usual convention with neural networks here in this work by Wolfgaang Mass he allows for direct wires from the input nodes to the output LTF gate. This flexibility ties in nicely with how we want to define a ReLU gate to be becoming removable under the random restrictions that we use.

Random Boolean functions vs any circuit class In everything that follows all samplings being done (denoted as $\sim$) are to be understood as sampling from an uniform distribution unless otherwise specified. Firstly we note this well-known lemma,

Claim 1. Let $f : \{-1,1\}^n \to \{-1,1\}$ be any given Boolean function. Then the following is true,

$$\mathbb{P}_{g\sim\{\{-1,1\}^n\to\{-1,1\}\}}\left[\mathbb{P}_{\mathbf{x}\sim\{-1,1\}^n}[f(\mathbf{x})=g(\mathbf{x})]\geq\frac{1}{2}+\epsilon\right]\leq e^{-2^{n+1}\epsilon^2}$$

From the above it follows that if N is the total number of functions in any circuit class (whose members be called C) then we have by union bound,

$$\mathbb{P}_{g\sim\{\{-1,1\}^n\to\{-1,1\}\}}\left[\exists C \text{ s.t } \mathbb{P}_{\mathbf{x}\sim\{-1,1\}^n}[C(\mathbf{x})=g(\mathbf{x})]\geq\frac{1}{2}+\epsilon\right]\leq Ne^{-2^{n+1}\epsilon^2} \tag{2.10}$$

Equipped with these basics we are now ready to begin the proof of the lowerbound against weight unrestricted LTF-of-ReLU circuits,

Proof of Theorem 2.5.4.

Definition 13. Let D denote arbitrary LTF-of-ReLU circuits over $\lfloor\log(\frac{n}{2})\rfloor$ bits.

For some $\frac{\epsilon}{3}\leq\frac{1}{2}$ and a size function denoted as $s(n,\epsilon)$ we use equation 2.10 , the definition of D above and the upperbound given earlier for the number of LTF-of-ReLU functions at a fixed circuit size (now used for circuits on $\lfloor\log(\frac{n}{2})\rfloor$ bits) to get,

$$\begin{aligned}\mathbb{P}_{f\sim\{0,1\}^{\lfloor\log(\frac{n}{2})\rfloor}\to\{0,1\}}&\left[\forall D \text{ s.t } |D|\leq s(n,\epsilon)\ |\mathbb{P}_{\mathbf{y}\sim\{0,1\}^{\lfloor\log(\frac{n}{2})\rfloor}}[f(\mathbf{y})=D(\mathbf{y})]\leq\left(\frac{1}{2}+\frac{\epsilon}{3}\right)\right]\\&\geq 1-2^{O(s^2\log^2(\frac{n}{2})\log(\log(\frac{n}{2})s))}e^{-\left(\frac{\epsilon^2}{9}\right)2^{1+\lfloor\log\left(\frac{n}{2}\right)\rfloor}}\\&\geq 1-2^{O(s^2k^2\log(ks))}e^{-\left(\frac{2\epsilon^2}{9}\right)2^k}\geq 1-e^{O(s^2k^2\log(ks))-\left(\frac{2\epsilon^2}{9}\right)2^k}\end{aligned}$$

whereby in the last inequality above we have assumed that $n=2^{k+1}$. This assumption is legitimate because we want to estimate certain large n asymptotics. Now for some $\theta>0$ if for large n we choose, $\epsilon>\sqrt{\frac{2\log^{2+\theta}(\frac{n}{2})}{n}}$ and $s=s(n,\epsilon)\leq O(\frac{\epsilon^{\frac{2}{2+\theta}}n^{\frac{1}{2+\theta}}}{2^{\frac{1}{2+\theta}}\log(\frac{n}{2})})$ then we have,

$$\begin{aligned}\mathbb{P}_{f\sim\{0,1\}^{\lfloor\log(\frac{n}{2})\rfloor}\to\{0,1\}}&\left[\forall D \text{ s.t } |D|\leq s(n,\epsilon)\ |\mathbb{P}_{\mathbf{y}\sim\{0,1\}^{\lfloor\log(\frac{n}{2})\rfloor}}[f(\mathbf{y})=D(\mathbf{y})]\leq\left(\frac{1}{2}+\frac{\epsilon}{3}\right)\right]\\&\geq 1-\frac{\epsilon}{3}\end{aligned} \tag{2.11}$$

Definition 14 ($\mathbf{F}^*$). Let F^* be the subset of all these f above for which the above event is true i.e

$$F^* := \left\{ f : \{0,1\}^{\lfloor \log(\frac{n}{2}) \rfloor} \to \{0,1\} \mid \forall D \text{ s.t } |D| \leq s(n,\epsilon) \; |\mathbb{P}_{\mathbf{y} \sim \{0,1\}^{\lfloor \log(\frac{n}{2}) \rfloor}}[f(\mathbf{y}) = D(\mathbf{y})] \right\}$$

Now we recall the definition of the Andreev function in equation 10 for the following definition and the claim,

Definition 15. Let ρ be a choice of a "restriction" whereby one is fixing all the input bits of A_n except 1 bit in each row of the matrix a. So the restricted function (call it $A_n|_\rho$) computes a function of the form,

$$A_n|_\rho : \{0,1\}^{\lfloor \log(\frac{n}{2}) \rfloor} \to \{0,1\}$$

Note that we shall henceforth be implicitly fixing a bijection mapping,
$\{0,1\}^n \to \{0,1\}^{\lfloor \frac{n}{2} \rfloor} \times \{0,1\}^{\lfloor \log(\frac{n}{2}) \rfloor \times \lfloor \frac{n}{2\lfloor \log(\frac{n}{2}) \rfloor} \rfloor}$ and hence for any function $C : \{0,1\}^n \to \{0,1\}$, it would be meaningful to talk of $C|_\rho$. From the definitions of A_n and ρ above, the following is immediate,

Claim 2. The truth table of $A_n|_\rho$ is the $\mathbf{x}$ string in the input to A_n that gets fixed by ρ. Thus we observe that if ρ is chosen uniformly at random then $A_n|_\rho$ is a $\lfloor \log(\frac{n}{2}) \rfloor$ bit Boolean function chosen uniformly at random.

Let f^* be any arbitrary member of F^*. Let $\mathbf{x}^* \in \{0,1\}^{\lfloor \frac{n}{2} \rfloor}$ be the truth-table of f^*. Let $\rho(\mathbf{x}^*)$ be restrictions on the input of A_n which fix the $\mathbf{x}$ part of its input to $\mathbf{x}^*$. So when we are sampling restrictions uniformly at random from the restrictions of the type $\rho(\mathbf{x}^*)$ these different instances differ in which bit of each row of the matrix a (of the input to A_n) they left unfixed and to what values did they fix the other entries of a. Let C be a n bit LTF-of-ReLU Boolean circuit of size say $w(n,\epsilon)$. Thus under a restriction of the type $\rho(\mathbf{x}^*)$ both C and A_n are $\lfloor \log(\frac{n}{2}) \rfloor$ bit Boolean functions.

Now we note that a ReLU gate over n bits upon a random restriction becomes redundant (and hence removable) iff its linear argument either reduces to a non-positive definite function or a positive definite function. In the former case the gate is computing the constant function zero and in the later case it is computing a linear function which can be simply implemented by introducing wires connecting the inputs directly to the output LTF gate. Thus in both the cases the resultant function no more needs the ReLU gate for it to be computed. (We note that such direct wires from the input

to the output gate were allowed in how the counting was done of the total number of LTF-of-ReLU Boolean functions at a fixed circuit size.) Combining both the cases we note that the conditions for collapse (in this sense) of a ReLU gate is identical to that of the conditions of collapse for a LTF gate for which Kane and Williams, 2016 in their Lemma 1.1 had proven the following,

Lemma 2.6.1 (Lemma 1.1 of Kane and Williams, 2016). Let $f : \{0,1\}^n \to \{0,1\}$ be a linear threshold function. Let $\mathscr{P}$ be a partition of $[n]$ into parts of equal size, and let $\mathscr{R}_{\mathscr{P}}$ be the distribution on restrictions $\rho : [n] \to \{0,1,*\}$ that randomly fixes all but one element of each part of $\mathscr{P}$. Then we have,

$$\mathbb{P}_{\rho \sim \mathscr{R}_{\mathscr{P}}} [f \text{ is not forced to a constant by } \rho] = O\left(\frac{|\mathscr{P}|}{\sqrt{n}}\right)$$

In our context the above implies,

$$\mathbb{P}_{\rho(\mathbf{x}^*)}[\text{ReLU}|_{\rho(\mathbf{x}^*)} \text{is removable }] \geq \eta$$

where $\eta = 1 - O(\frac{\log n}{\sqrt{n}})$

The above definition of η implies,

$$\begin{aligned}
&\mathbb{P}_{\rho(\mathbf{x}^*)}[\text{ A } n-\text{bit ReLU is } \textit{not} \text{ forced to a constant }] \leq 1-\eta \\
\implies &\mathbb{E}_{\rho(\mathbf{x}^*)}[\text{ Number of ReLUs of C } \textit{not} \text{ forced to a constant }] \leq w(n,\epsilon)(1-\eta) \\
\implies &\mathbb{P}_{\rho(\mathbf{x}^*)}[\text{ Number of ReLUs of C } \textit{not} \text{ forced to a constant } > s(n,\epsilon)] \\
&\quad \leq \frac{\mathbb{E}_{\rho(\mathbf{x}^*)}[\text{ Number of ReLUs of C } \textit{not} \text{ forced to a constant }]}{s(n,\epsilon)} \\
\implies &\mathbb{P}_{\rho(\mathbf{x}^*)}[\text{ Number of ReLUs of C } \textit{not} \text{ forced to a constant } \geq s(n,\epsilon)] \leq \frac{w(n,\epsilon)(1-\eta)}{s(n,\epsilon)} \\
\implies &\mathbb{P}_{\rho(\mathbf{x}^*)}[\text{ Size of } C|_{\rho(\mathbf{x}^*)} \leq s(n,\epsilon)] \geq 1 - \frac{w(n,\epsilon)(1-\eta)}{s(n,\epsilon)}
\end{aligned} \tag{2.12}$$

Now we compare with the definitions of ϵ and f^* to observe that (a) with probability at least $1 - \frac{w(n,\epsilon)(1-\eta)}{s(n,\epsilon)}$, $C|_{\rho(\mathbf{x}^*)}$ is a circuit of the type called "D" in the event in equation 2.11 and (b) by definition of the Andreev function it follows that $A_n|_{\rho(\mathbf{x}^*)}$ has its truth table given by $\mathbf{x}^*$ and hence it specifies the same function as $f^* \in F^*$. Hence $\forall \mathbf{x}^*$ and $\rho(\mathbf{x}^*)$ we can read off from equation 2.11,

$$\mathbb{P}_{\mathbf{y}\sim\{0,1\}^{\lfloor\log(\frac{n}{2})\rfloor}}[C|_{\rho(\mathbf{x}^*)}(\mathbf{y}) = A_n|_{\rho(\mathbf{x}^*)}(\mathbf{y})| \text{ Size of } C|_{\rho(\mathbf{x}^*)} \leq s(n,\epsilon)] \leq \frac{1}{2} + \frac{\epsilon}{3} \tag{2.13}$$

Recalling Definition 14, the equation 2.11 can be written as,

$$\mathbb{P}_{f\sim\{0,1\}^{\lfloor\log(\frac{n}{2})\rfloor}\to\{0,1\}}[f \in F^*] \geq 1 - \frac{\epsilon}{3} \tag{2.14}$$

Claim 3. Circuits C have low correlation with the Andreev function

$$\mathbb{P}_{\mathbf{z}\sim\{0,1\}^n}[C(z) = A_n(z)] \leq \frac{\epsilon}{3} + \frac{w(n,\epsilon)(1-\eta)}{s(n,\epsilon)} + \frac{1}{2} + \frac{\epsilon}{3}$$

Proof. We think of sampling a $z \sim \{0,1\}^n$ as a two step process of first sampling a $\tilde{f}$, a $\lfloor\log(\frac{n}{2})\rfloor$ bit Boolean function and fixing the first $\lfloor\frac{n}{2}\rfloor$ bits of z to be the truth-table of $\tilde{f}$ and then we randomly assign values to the remaining $\lfloor\frac{n}{2}\rfloor$ bits of z. Call these later $\lfloor\frac{n}{2}\rfloor$ bit string to be $\mathbf{x}_{other}$.

$$\begin{aligned}
\mathbb{P}_{\mathbf{z}\sim\{0,1\}^n}[C(z) = A_n(z)] &= \mathbb{P}_{\mathbf{z}\sim\{0,1\}^n}[(C(z) = A_n(z)) \cap (\tilde{f} \in F^*)] + \mathbb{P}_{\mathbf{z}\sim\{0,1\}^n}[(C(z) = A_n(z)) \cap (\tilde{f} \notin F^*)] \\
&= \mathbb{P}_{\mathbf{z}\sim\{0,1\}^n}[(C(z) = A_n(z)) \mid (\tilde{f} \in F^*)]\mathbb{P}_{\mathbf{z}\sim\{0,1\}^n}[\tilde{f} \in F^*] \\
&+ \mathbb{P}_{\mathbf{z}\sim\{0,1\}^n}[(C(z) = A_n(z)) \cap (\tilde{f} \notin F^*)] \\
&\leq \mathbb{P}_{\mathbf{z}\sim\{0,1\}^n}[(C(z) = A_n(z)) \mid (\tilde{f} \in F^*)] + \mathbb{P}_{\mathbf{z}\sim\{0,1\}^n}[\tilde{f} \notin F^*] \\
&\leq \mathbb{P}_{\mathbf{z}\sim\{0,1\}^n}[(C(z) = A_n(z)) \mid (\tilde{f} \in F^*)] + \frac{\epsilon}{3}
\end{aligned}$$

In the last line above we have invoked equation 2.14. Now we note that sampling the n bit string z such that $\tilde{f} \in F^*$ is the same as doing a random restriction of the type $\rho(\tilde{f})$ and then randomly picking a $\lfloor\log(\frac{n}{2})\rfloor$ bit string say $\mathbf{y}$. So we can rewrite the last inequality as,

$$
\begin{aligned}
\mathbb{P}_{\mathbf{z}\sim\{0,1\}^n}[C(z) = A_n(z)] &\leq \mathbb{P}_{(\rho(\tilde{f}),\mathbf{y})}[C(\rho(\tilde{f}),\mathbf{y}) = A_n(\rho(\tilde{f}),\mathbf{y})] + \frac{\epsilon}{3} \\
&\leq \mathbb{E}_{(\rho(\tilde{f}),\mathbf{y})}[\mathbf{1}_{C(\rho(\tilde{f}),\mathbf{y})=A_n(\rho(\tilde{f}),\mathbf{y})} \mid (\tilde{f} \in F^*)] + \frac{\epsilon}{3} \\
&\leq \mathbb{E}_{(\rho(\tilde{f}),\mathbf{y})}[\mathbf{1}_{C(\rho(\tilde{f}),\mathbf{y})=A_n(\rho(\tilde{f}),\mathbf{y})}\mathbf{1}_{\text{Size of } C|_{\rho(\tilde{f})}<s(n,\epsilon)} \mid (\tilde{f} \in F^*)] \\
&+ \mathbb{E}_{(\rho(\tilde{f}),\mathbf{y})}[\mathbf{1}_{C(\rho(\tilde{f}),\mathbf{y})=A_n(\rho(\tilde{f}),\mathbf{y})}\mathbf{1}_{\text{Size of } C|_{\rho(\tilde{f})}\geq s(n,\epsilon)} \mid (\tilde{f} \in F^*)] + \frac{\epsilon}{3} \\
&\leq \mathbb{P}_{(\rho(\tilde{f}),\mathbf{y})}[C(\rho(\tilde{f}),\mathbf{y}) = A_n(\rho(\tilde{f}),\mathbf{y}) \mid \Big((\text{Size of } C|_{\rho(\tilde{f})} < s(n,\epsilon)) \cap (\tilde{f} \in F^*)\Big)] \\
&+ \mathbb{P}_{(\rho(\tilde{f}),\mathbf{y})}[\text{Size of } C|_{\rho(\tilde{f})} \geq s(n,\epsilon) \mid (\tilde{f} \in F^*)] + \frac{\epsilon}{3} \\
&\leq \left(\frac{1}{2} + \frac{\epsilon}{3}\right) + \frac{w(n,\epsilon)(1-\eta)}{s(n,\epsilon)} + \frac{\epsilon}{3}
\end{aligned}
$$

In the last step above we have used equations 2.13 and 2.12. □

So after putting back the values of η and the largest scaling of $s(n,\epsilon)$ that we can have (from equation 2.11), the upperbound on the above probability becomes,

$$\frac{1}{2} + \frac{2\epsilon}{3} + O\left(\frac{w(n,\epsilon)\log(n)}{\sqrt{n}(\frac{\epsilon^{\frac{2}{2+\theta}} n^{\frac{1}{2+\theta}}}{2^{\frac{1}{2+\theta}}\log(\frac{n}{2})})}\right)$$

Thus the probability is upperbounded by $\frac{1}{2} + \epsilon$ as long as $w(n,\epsilon) = O\left(\frac{\epsilon^{1+\frac{2}{2+\theta}} n^{\frac{1}{2}+\frac{1}{2+\theta}} \log\left(\frac{n}{2}\right)}{\log(n)}\right)$

Stated as a lowerbound we have that if a LTF-of-ReLU has to match the $n-$bit Andreev function on more than $\frac{1}{2} + \epsilon$ fraction of the inputs for $\epsilon > \sqrt{\frac{2\log^{2+\theta}(\frac{n}{2})}{n}}$ for some $\theta > 0$ (asymptotically this is like having a constant ϵ) then the LTF-of-ReLU needs to be of size $\Omega(\epsilon^{\frac{4+\theta}{2+\theta}} n^{\frac{1}{2}+\frac{1}{2+\theta}})$. Now we define $\delta \in (0, \frac{1}{2})$ such that $\delta = \frac{\theta}{2(2+\theta)}$ and that gives the form of the almost linear lowerbound as stated in the theorem. □

Appendix To Chapter 2

2.A Expressing piecewise linear functions using ReLU DNNs

Proof of Theorem 2.2.2. Any continuous piecewise linear function $\mathbb{R} \to \mathbb{R}$ which has m pieces can be specified by three pieces of information, (1) s_L the slope of the left most piece, (2) the coordinates of the non-differentiable points specified by a $(m-1)-$tuple $\{(a_i, b_i)\}_{i=1}^{m-1}$ (indexed from left to right) and (3) s_R the slope of the rightmost piece. A tuple $(s_L, s_R, (a_1, b_1), \ldots, (a_{m-1}, b_{m-1})$ uniquely specifies a m piecewise linear function from $\mathbb{R} \to \mathbb{R}$ and vice versa. Given such a tuple, we construct a 2-layer DNN which computes the same piecewise linear function.

One notes that for any $a, r \in \mathbb{R}$, the function

$$f(x) = \begin{cases} 0 & x \leq a \\ r(x-a) & x > a \end{cases} \tag{2.15}$$

is equal to $\operatorname{sgn}(r) \max\{|r|(x-a), 0\}$, which can be implemented by a 2-layer ReLU DNN with size 1. Similarly, any function of the form,

$$g(x) = \begin{cases} t(x-a) & x \leq a \\ 0 & x > a \end{cases} \tag{2.16}$$

is equal to $-\operatorname{sgn}(t) \max\{-|t|(x-a), 0\}$, which can be implemented by a 2-layer ReLU DNN with size 1. The parameters r, t will be called the *slopes* of the function, and a will be called the *breakpoint* of the function.

If we can write the given piecewise linear function as a sum of m functions of the form (2.15) and (2.16), then by Lemma 2.A.3 we would be done. It turns out that such a decomposition of any p piece PWL function $h : \mathbb{R} \to \mathbb{R}$ as a sum of p flaps can always be arranged where the breakpoints of the p flaps all are all contained in the $p-1$ breakpoints of h. First, observe that adding a constant to a function

does not change the complexity of the ReLU DNN expressing it, since this corresponds to a bias on the output node. Thus, we will assume that the value of h at the last break point a_{m-1} is $b_{m-1} = 0$.

We now use a single function f of the form (2.15) with slope r and breakpoint $a = a_{m-1}$, and $m-1$ functions $g_1, \ldots, g_{m-1}$ of the form (2.16) with slopes $t_1, \ldots, t_{m-1}$ and breakpoints $a_1, \ldots, a_{m-1}$, respectively.

Thus, we wish to express $h = f + g_1 + \ldots + g_{m-1}$. Such a decomposition of h would be valid if we can find values for $r, t_1, \ldots, t_{m-1}$ such that (1) the slope of the above sum is $= s_L$ for $x < a_1$, (2) the slope of the above sum is $= s_R$ for $x > a_{m-1}$, and (3) for each $i \in \{1, 2, 3, .., m-1\}$ we have $b_i = f(a_i) + g_1(a_i) + \ldots + g_{m-1}(a_i)$.

The above corresponds to asking for the existence of a solution to the following set of simultaneous linear equations in $r, t_1, \ldots, t_{m-1}$:

$$s_R = r, \;\; s_L = t_1 + t_2 + \ldots + t_{m-1}, \;\; b_i = \sum_{j=i+1}^{m-1} t_j(a_{j-1} - a_j) \text{ for all } i = 1, \ldots, m-2$$

It is easy to verify that the above set of simultaneous linear equations has a unique solution. Indeed, r must equal s_R, and then one can solve for $t_1, \ldots, t_{m-1}$ starting from the last equation $b_{m-2} = t_{m-1}(a_{m-2} - a_{m-1})$ and then back substitute to compute $t_{m-2}, t_{m-3}, \ldots, t_1$.

The lower bound of $p-1$ on the size for any 2-layer ReLU DNN that expresses a p piece function follows from Lemma 2.A.7. □

One can do better in terms of size when the rightmost piece of the given function is flat, i.e., $s_R = 0$. In this case $r = 0$, which means that $f = 0$; thus, the decomposition of h above is of size $p-1$. A similar construction can be done when $s_L = 0$. This gives the following statement which will be useful for constructing our forthcoming hard functions.

Corollary 2.A.1. If the rightmost or leftmost piece of a $\mathbb{R} \to \mathbb{R}$ piecewise linear function has 0 slope, then we can compute such a p piece function using a 2-layer DNN with size $p-1$.

Proof of theorem 2.2.3. Since any piecewise linear function $\mathbb{R}^n \to \mathbb{R}$ is representable by a ReLU DNN by Corollary 2.2.1, the proof simply follows from the fact that the family of continuous piecewise linear functions is dense in any $L^p(\mathbb{R}^n)$ space, for $1 \leq p \leq \infty$. □

Now we will collect some straightforward observations that will be used often in constructing complex neural functions starting from simple ones. The following operations preserve the property of being representable by a ReLU DNN.

Lemma 2.A.2. [Function Composition] If $f_1 : \mathbb{R}^d \to \mathbb{R}^m$ is represented by a d, m ReLU DNN with depth $k_1 + 1$ and size s_1, and $f_2 : \mathbb{R}^m \to \mathbb{R}^n$ is represented by an m, n ReLU DNN with depth $k_2 + 1$ and size s_2, then $f_2 \circ f_1$ can be represented by a d, n ReLU DNN with depth $k_1 + k_2 + 1$ and size $s_1 + s_2$.

Proof. Follows from (1.1) and the fact that a composition of affine transformations is another affine transformation. □

Lemma 2.A.3. [Function Addition] If $f_1 : \mathbb{R}^n \to \mathbb{R}^m$ is represented by a n, m ReLU DNN with depth $k + 1$ and size s_1, and $f_2 : \mathbb{R}^n \to \mathbb{R}^m$ is represented by a n, m ReLU DNN with depth $k + 1$ and size s_2, then $f_1 + f_2$ can be represented by a n, m ReLU DNN with depth $k + 1$ and size $s_1 + s_2$.

Proof. We simply put the two ReLU DNNs in parallel and combine the appropriate coordinates of the outputs. □

Lemma 2.A.4. [Taking maximums/minimums] Let $f_1, \ldots, f_m : \mathbb{R}^n \to \mathbb{R}$ be functions that can each be represented by $\mathbb{R}^n \to \mathbb{R}$ ReLU DNNs with depths $k_i + 1$ and size s_i, $i = 1, \ldots, m$. Then the function $f : \mathbb{R}^n \to \mathbb{R}$ defined as $f(\mathbf{x}) := \max\{f_1(\mathbf{x}), \ldots, f_m(\mathbf{x})\}$ can be represented by a ReLU DNN of depth at most $\max\{k_1, \ldots, k_m\} + \log(m) + 1$ and size at most $s_1 + \ldots s_m + 4(2m - 1)$. Similarly, the function $g(\mathbf{x}) := \min\{f_1(\mathbf{x}), \ldots, f_m(\mathbf{x})\}$ can be represented by a ReLU DNN of depth at most $\max\{k_1, \ldots, k_m\} + \lceil \log(m) \rceil + 1$ and size at most $s_1 + \ldots s_m + 4(2m - 1)$.

Proof. We prove this by induction on m. The base case $m = 1$ is trivial. For $m \geq 2$, consider $g_1 := \max\{f_1, \ldots, f_{\lfloor \frac{m}{2} \rfloor}\}$ and $g_2 := \max\{f_{\lfloor \frac{m}{2} \rfloor + 1}, \ldots, f_m\}$. By the induction hypothesis (since $\lfloor \frac{m}{2} \rfloor, \lceil \frac{m}{2} \rceil < m$ when $m \geq 2$), g_1 and g_2 can be represented by ReLU DNNs of depths at most $\max\{k_1, \ldots, k_{\lfloor \frac{m}{2} \rfloor}\} + \lceil \log(\lfloor \frac{m}{2} \rfloor) \rceil + 1$ and $\max\{k_{\lfloor \frac{m}{2} \rfloor + 1}, \ldots, k_m\} + \lceil \log(\lceil \frac{m}{2} \rceil) \rceil + 1$ respectively, and sizes at most $s_1 + \ldots s_{\lfloor \frac{m}{2} \rfloor} + 4(2\lfloor \frac{m}{2} \rfloor - 1)$ and $s_{\lfloor \frac{m}{2} \rfloor + 1} + \ldots + s_m + 4(2\lfloor \frac{m}{2} \rfloor - 1)$, respectively. Therefore, the function $G : \mathbb{R}^n \to \mathbb{R}^2$ given by $G(\mathbf{x}) = (g_1(\mathbf{x}), g_2(\mathbf{x}))$ can be implemented by a ReLU DNN with depth at most $\max\{k_1, \ldots, k_m\} + \lceil \log(\lceil \frac{m}{2} \rceil) \rceil + 1$ and size at most $s_1 + \ldots + s_m + 4(2m - 2)$.

We now show how to represent the function $T : \mathbb{R}^2 \to \mathbb{R}$ defined as $T(x,y) = \max\{x,y\} = \frac{x+y}{2} + \frac{|x-y|}{2}$ by a 2-layer ReLU DNN with size 4 – see Figure 2.A.1. The result now follows from the fact that $f = T \circ G$ and Lemma 2.A.2. □

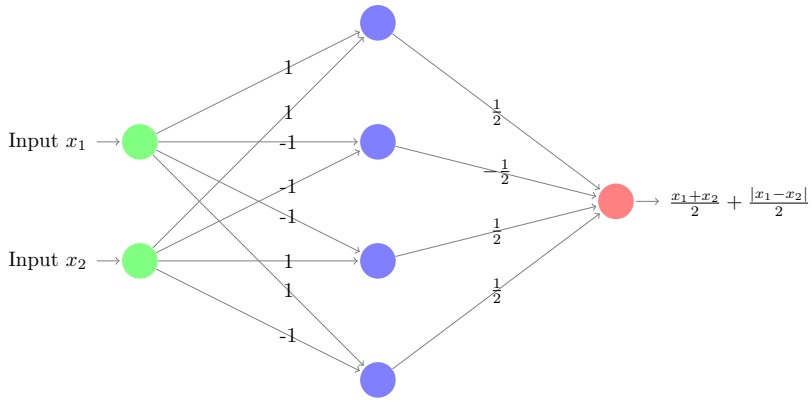


FIGURE 2.A.1: A 2-layer ReLU DNN computing $\max\{x_1, x_2\} = \frac{x_1+x_2}{2} + \frac{|x_1-x_2|}{2}$

Lemma 2.A.5. Any affine transformation $T : \mathbb{R}^n \to \mathbb{R}^m$ is representable by a 2-layer ReLU DNN of size $2m$.

Proof. Simply use the fact that $T = (I \circ \sigma \circ T) + (-I \circ \sigma \circ (-T))$, and the right hand side can be represented by a 2-layer ReLU DNN of size $2m$ using Lemma 2.A.3. □

Lemma 2.A.6. Let $f : \mathbb{R} \to \mathbb{R}$ be a function represented by a $\mathbb{R} \to \mathbb{R}$ ReLU DNN with depth $k+1$ and widths $w_1, \ldots, w_k$ of the k hidden layers. Then f is a PWL function with at most $2^{k-1} \cdot (w_1 + 1) \cdot w_2 \cdot \ldots \cdot w_k$ pieces.

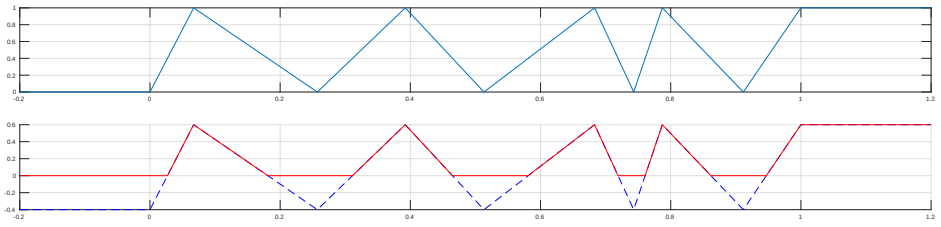


FIGURE 2.A.2: The number of pieces increasing after activation. If the blue function is f, then the red function $g = \max\{0, f + b\}$ has at most twice the number of pieces as f for any bias $b \in \mathbb{R}$.

Proof. We prove this by induction on k. The base case is $k = 1$, i.e, we have a 2-layer ReLU DNN. Since every activation node can produce at most one breakpoint in the piecewise linear function, we can get at most w_1 breakpoints, i.e., $w_1 + 1$ pieces.

Now for the induction step, assume that for some $k \geq 1$, any $\mathbb{R} \to \mathbb{R}$ ReLU DNN with depth $k+1$ and widths $w_1, \ldots, w_k$ of the k hidden layers produces at most $2^{k-1} \cdot (w_1 + 1) \cdot w_2 \cdot \ldots \cdot w_k$ pieces.

Consider any $\mathbb{R} \to \mathbb{R}$ ReLU DNN with depth $k+2$ and widths $w_1, \ldots, w_{k+1}$ of the $k+1$ hidden layers. Observe that the input to any node in the last layer is the output of a $\mathbb{R} \to \mathbb{R}$ ReLU DNN with depth $k+1$ and widths $w_1, \ldots, w_k$. By the induction hypothesis, the input to this node in the last layer is a piecewise linear function f with at most $2^{k-1} \cdot (w_1 + 1) \cdot w_2 \cdot \ldots \cdot w_k$ pieces. When we apply the activation, the new function $g(x) = \max\{0, f(x)\}$, which is the output of this node, may have at most twice the number of pieces as f, because each original piece may be intersected by the x-axis; see Figure 2.A.2. Thus, after going through the layer, we take an affine combination of w_{k+1} functions, each with at most $2 \cdot (2^{k-1} \cdot (w_1 + 1) \cdot w_2 \cdot \ldots \cdot w_k)$ pieces. In all, we can therefore get at most $2 \cdot (2^{k-1} \cdot (w_1 + 1) \cdot w_2 \cdot \ldots \cdot w_k) \cdot w_{k+1}$ pieces, which is equal to $2^k \cdot (w_1 + 1) \cdot w_2 \cdot \ldots \cdot w_k \cdot w_{k+1}$, and the induction step is completed. □

Lemma 2.A.6 has the following consequence about the depth and size tradeoffs for expressing functions with agiven number of pieces.

Lemma 2.A.7. Let $f : \mathbb{R} \to \mathbb{R}$ be a piecewise linear function with p pieces. If f is represented by a ReLU DNN with depth $k+1$, then it must have size at least $\frac{1}{2}kp^{1/k} - 1$. Conversely, any piecewise linear function f that is represented by a ReLU DNN of depth $k+1$ and size at most s, can have at most $(\frac{2s}{k})^k$ pieces.

Proof. Let widths of the k hidden layers be $w_1, \ldots, w_k$. By Lemma 2.A.6, we must have

$$2^{k-1} \cdot (w_1 + 1) \cdot w_2 \cdot \ldots \cdot w_k \geq p. \tag{2.17}$$

By the AM-GM inequality, minimizing the size $w_1 + w_2 + \ldots + w_k$ subject to (2.17), means setting $w_1 + 1 = w_2 = \ldots = w_k$. This implies that $w_1 + 1 = w_2 = \ldots = w_k \geq \frac{1}{2}p^{1/k}$. The first statement follows. The second statement follows using the AM-GM inequality again, this time with a restriction on $w_1 + w_2 + \ldots + w_k$. □

2.B Proof of Proposition 2.5.2

We first observe that the set of points where $\max\{0, x_1, x_2\}$ is not differentiable is precisely the union of the three half-lines (or rays) $\{(x_1, x_2) : x_1 = x_2, x_1 \geq 0\} \cup \{(0, x_2) : x_2 \leq 0\} \cup \{(x_1, 0) : x_1 \leq 0\}$. On the other hand, consider any Sum-of-ReLU circuit, which can be expressed as a function of the

form

$$f(x) = \sum_{i=1}^{w} c_i \max\{0, \langle a^i, x\rangle + b_i\},$$

where $w \in \mathbb{N}$ is the number of ReLU gates in the ciruit, and $a^i \in \mathbb{R}^2, b_i, c_i \in \mathbb{R}$ for all $i = 1, \ldots, w$. This implies that $f(x)$ is piecewise linear and the set of points where $f(x)$ is not differentiable is *precisely* the union of the w lines $\langle a^i, x\rangle + b_i = 0$, $i = 1, \ldots, w$. Since a union of lines cannot equal the union of the three half-lines $\{(x_1, x_2) : x_1 = x_2, x_1 \geq 0\} \cup \{(0, x_2) : x_2 \leq 0\} \cup \{(x_1, 0) : x_1 \leq 0\}$, we obtain the consequence that $\max\{0, x_1, x_2\}$ cannot be represented by a Sum-of-ReLU circuit, no matter how many ReLU gates are used.

2.C Simulating an LTF gate by a ReLU gate

Claim 4. Any LTF gate $\{-1, 1\}^n \to \{-1, 1\}$ can be simulated by a Sum-of-ReLU circuit with at most 2 ReLU gates.

Proof. Given a LTF gate $(2\mathbf{1}_{\langle a,x\rangle + b \geq 0} - 1)$ it separates the points in $\{-1, 1\}^n$ into two subsets such that the plane $\langle a, x\rangle + b = 0$ is a separating hyperplane between the two sets. Let $-p < 0$ be the value of the function $\langle a, x\rangle + b$ at that hypercube vertex on the "-1" side which is closest to this separating plane. Now imagine a continuous piecewise linear function $f : \mathbb{R} \to \mathbb{R}$ such that $f(x) = -1$ for $x \leq -p$, $f(x) = 1$ for $x \geq 0$ and for $x \in (-p, 0)$ f is the straight line function connecting $(-p, -1)$ to $(0, 1)$. It follows from Theorem 2.2.2 that this f can be implemented by a $\mathbb{R} \to \mathbb{R}$ Sum-of-ReLU with at most 2 ReLU gates hinged at the points $-p$ and 0 on the domain. Because the affine transformation $\langle a, x\rangle + b$ can be implemented by the wires connecting the n input nodes to the layer of ReLUs it follows that there exists a $\mathbb{R}^n \to \mathbb{R}$ Sum-of-ReLU with at most 2 ReLU gates implementing the function $g(x) = f(\langle a, x\rangle + b) : \mathbb{R}^n \to \mathbb{R}$. Its clear that $g(\mathbf{x}) = \text{LTF}(\mathbf{x})$ for all $\mathbf{x} \in \{-1, 1\}^n$. □

2.D PARITY on $k-$bits can be implemented by a $O(k)$ Sum-of-ReLU circuit

For this proof its convenient to think of the PARITY function as the following map,

$$\begin{aligned} \text{PARITY} : \{0, 1\}^k &\to \{0, 1\} \\ \mathbf{x} &\mapsto \left(\sum_{i=1}^{k} x_i\right) \mod 2 \end{aligned}$$

Its clear that that in the evaluation of the PARITY function as stated above the required sum over the coordinates of the input Boolean vector will take as value every integer in the set, $\{0, 1, 2, .., k\}$. The PARITY function can then be lifted to a $f : \mathbb{R} \to \mathbb{R}$ function such that, $f(y) = 0$ for all $y \leq 0$, $f(y) = y \mod 2$ for all $y \in 1, 2, .., k$, $f(y) = k \mod 2$ for all $y > k$ and for any $y \in (p, p+1)$ for $p \in \{0, 1, .., k-1\}$ f is the straight line function connecting the points, $(p, p \mod 2)$ and $(p+1, (p+1) \mod 2)$. Thus f is a continuous piecewise linear function on $\mathbb{R}$ with $k+2$ linear pieces. Then it follows from Theorem 2.2.3 that this f can be implemented by a $\mathbb{R} \to \mathbb{R}$ Sum-of-ReLU circuit with at most $k+1$ ReLU gates hinged at the points $\{0, 1, 2, .., k\}$ on the domain. The wires from the k inputs of the ReLU gates can implement the linear function $\sum_{i=1}^{k} x_i$. Thus it follows that there exists a $\mathbb{R}^k \to \mathbb{R}$ Sum-of-ReLU circuit (say C) such that, $\mathrm{C}(\mathbf{x}) = \mathrm{PARITY}(\mathbf{x})$ for all $\mathbf{x} \in \{0, 1\}^k$.

2.E Proof of Theorem 2.5.5 (Proving smallness of the sign-rank of LTF-of-(ReLU)$^{\mathbf{d}-1}$ with weight restrictions only on the bottom most layer)

For a $\{-1, 1\}^M \to \{-1, 1\}$ LTF-of-ReLU circuit with any given weights on the network the inputs to the threshold function of the top LTF gate are some set of 2^M real numbers (one for each input). Over all these inputs let $p > 0$ be the distance from 0 of the largest negative number on which the LTF gate ever gets evaluated. Then by increasing the bias at this last LTF gate by a quantity less then p we can ensure that no input to this LTF gate is 0 while the entire circuit still computes the same Boolean function as originally. So we can assume without loss of generality that the input to the threshold function at the top LTF gate is never 0. We also recall that the weights at the bottom most layer are constrained to be integers of magnitude at most $W > 0$.

Let this depth d LTF-of-(ReLU)$^{d-1}$ circuit map $\{-1, 1\}^m \times \{-1, 1\}^m \to \{-1, 1\}$. Let $\{w_k\}_{k=1}^{d-1}$ be the widths of the ReLU layers at depths indexed by increasing k with increasing distance from the input. Thus, the output LTF gate gets w_{d-1} inputs; the j-th input, for $j = 1, 2, .., w_{d-1}$, is the output of a circuit C_j of depth $d-1$ composed of only ReLU gates. Let $f_j(\mathbf{x}, \mathbf{y}) : \{-1, 1\}^m \times \{-1, 1\}^m \to \mathbb{R}$ be the pseudo-Boolean function implemented by C_j.

Thus the output of the overall LTF-of-(ReLU)$^{d-1}$ circuit is,

$$f(\mathbf{x},\mathbf{y}) := \text{LTF}\left[\beta + \sum_{j=1}^{w_{d-1}} \alpha_j f_j(\mathbf{x},\mathbf{y})\right] \tag{2.18}$$

Lemma 2.E.1. Let $k \geq 1$ and $w_1, \ldots, w_k \geq 1$ be natural numbers. Consider a family of depth $k+1$ circuits (say indexed by $i \in I$ for some index set I) with $2m$ inputs and a single output and consisting of only ReLU gates. Let all of them have w_j ReLU gates at depth j, with $j = 1$ corresponding to the layer closest to the input (note that single output ReLU gate is not counted here). Moreover, let all of the circuits in the family have the same weights in all their layers except for the layer closest to the output. We restrict the inputs to $\{-1,1\}^m \times \{-1,1\}^m$ and let the i^{th} circuit ($i \in I$) implement a pseudo-Boolean function $g_i : \{-1,1\}^m \times \{-1,1\}^m \to \mathbb{R}$. Assume that the weights of the w_1 ReLU gates in the layer closest to the input are restricted as per Definition 12. For every $i \in I$, define the $2^m \times 2^m$ matrix $G_i(\mathbf{x},\mathbf{y})$ whose rows are indexed by $\mathbf{x} \in \{-1,1\}^m$ and columns are indexed by $\mathbf{y} \in \{-1,1\}^m$ as follows:

$$G_i(\mathbf{x},\mathbf{y}) = g_i(\mathbf{x},\mathbf{y}).$$

Then there exists a fixed way to order the rows and columns such that for each G_i there exists a *contiguous* partitioning (which can depend on i) of its rows and columns into $O\big((\prod_{i=1}^{k} w_i)(mW)\big)$ blocks (thus, G_i has $O\big((\prod_{i=1}^{k} w_i)^2(mW)^2\big)$ blocks), and within each block G_i is constant valued.

Before we prove the above lemma, let us see why it implies Theorem 2.5.5.

Proof. (of Theorem 2.5.5) Let $F_j(\mathbf{x},\mathbf{y})$ be the matrix obtained from the ReLU circuit outputs $f_j(\mathbf{x},\mathbf{y})$ from (2.18), and let $F(\mathbf{x},\mathbf{y})$ be the matrix obtained from $f(\mathbf{x},\mathbf{y})$. Let $J_{2^m \times 2^m}$ be the matrix of all ones. Then

$$\begin{aligned}\text{sign-rank}(F(\mathbf{x},\mathbf{y})) &= \text{sign-rank}\left(\text{sign}\left[\beta J_{2^m\times 2^m} + \sum_{j=1}^{w_{d-1}} \alpha_j F_j(\mathbf{x},\mathbf{y})\right]\right)\\ &\leq \text{rank}\left(\beta J_{2^m\times 2^m} + \sum_{j=1}^{w_{d-1}} \alpha_j F_j(\mathbf{x},\mathbf{y})\right)\\ &\leq 1 + \sum_{j=1}^{w_{d-1}} \text{rank}(F_j(\mathbf{x},\mathbf{y}))\\ &= O\left(\left(\prod_{k=1}^{d-1} w_k\right)^2 (mW)^2\right)\end{aligned}$$

where the first inequality follows from the definition of sign-rank, the second inequality follows from the subadditivty of rank and the last inequality is a consequence of using Lemma 2.E.1 at depth $k+1 = d-1$. Indeed, a matrix with block structure as in the conclusion of Lemma 2.E.1 has rank at most $O\big((\prod_{i=1}^{k} w_i)^2 (mW)^2\big)$ by expressing it as a sum of these many matrices of rank one and using subaddivity of rank.

Now we recall that the Chattopadhyay-Mande function g (which is linear sized depth 2 LTF) on $2m = 2(n^{\frac{4}{3}} + n\log n)$ bits has sign-rank $\Omega(2^{\frac{n^{\frac{1}{3}}}{81}})$. It follows that we can find a constant $C > 1$ s.t for all large enough n we have, $C^4 n^{\frac{4}{3}} \geq m$. Then we would have, sign-rank$(g) = \Omega(2^{\frac{m^{\frac{1}{4}}}{81C}})$. From the above upper bound on the sign-rank of our bottom layer weight restricted LTF-of-(ReLU)$^{d-1}$ with widths $\{w_k\}_{k=1}^{d-1}$ it follows that for this to represent this Chattopadhyay-Mande function it would need, $\left(\left(\prod_{k=1}^{d-1} w_k\right)^2 (mW)^2\right) = \Omega(2^{\frac{m^{\frac{1}{4}}}{81C}})$. Hence it follows by the "AM≥GM" inequality that the size $(1 + \sum_{k=1}^{d-1} w_i)$ required for such LTF-of-(ReLU)$^{d-1}$ circuits to represent the Chattopadhyay-Mande function is $\Omega\left((d-1)\left[\frac{2^{m^{\frac{1}{8}}}}{mW}\right]^{\frac{1}{(d-1)}}\right)$.

The statement about LTF circuits is a straightforward consequence of the above result and Claim 4 in Appendix 2.C which says that any LTF gate can be simulated by 2 ReLU gates. □

Towards proving Lemma 2.E.1 we first make the following observation,

Claim 5. Let w, M, D be fixed natural numbers. Let $A_1, \ldots, A_w$ be any $M \times M$ matrices such that there exists a fixed way to order the rows and columns for each of the A_i such that they get partitioned contiguously into D blocks (not necessarily equal in size) and this partitioning is such that A_i is constant valued within each of the D^2 blocks. Then $A := A_1 + \ldots + A_w$ is an $M \times M$ matrix whose

rows and columns can be partitioned contiguously into $w(D-1)+1$ groups such that A is constant valued within each block defined by this partition of the rows and columns.

Proof. The partition of the rows of A_i into D contiguous blocks is equivalent to a choice of $D-1$ lines out of $M-1$ lines. (Potentially a different set of $D-1$ lines for each A_i) But the guarantee that this partitioning is induced in each of the A_i by the same ordering of the rows means that When we sum the matrices, the refined partition in the sum corresponds to some selection of $w(D-1)$ lines out of the $M-1$ lines. This gives us at most $w(D-1)+1$ contiguous blocks among the rows of the sum matrix. The same argument holds for the columns. □

Proof of Lemma 2.E.1. We will prove this Lemma by induction on k.

The base case of the induction $k=1$ (i.e depth 2) A single ReLU gate in the bottom most layer of the net which receives a tuple of vectors $(\mathbf{x}, \mathbf{y})$ as input gives as output the number, $\max\{0, \langle \mathbf{a}^1, \mathbf{x}\rangle + \langle \mathbf{a}^2, \mathbf{y}\rangle + b\}$, for some $\mathbf{a}^1, \mathbf{a}^2 \in \mathbb{R}^m$ and $b \in \mathbb{R}$. Since the entries of $\mathbf{a}^1, \mathbf{a}^2$ and b are assumed to be integers bounded by $W > 0$ and $\mathbf{x}, \mathbf{y} \in \{-1, 1\}^m$, the terms $\langle \mathbf{a}^1, \mathbf{x}\rangle$ and $\langle \mathbf{a}^2, \mathbf{y}\rangle$ can each take at most $O(mW)$ different values. So we can arrange the rows and columns of the $2^m \times 2^m$ dimensional output matrix of this gate in increasing order of $\langle \mathbf{a}^1, \mathbf{x}\rangle$ and $\langle \mathbf{a}^2, \mathbf{y}\rangle$ and then partition the rows and columns contiguously according to these values. And we note that because of the weight restriction as in Definition 12 that applies to each of ReLU gates in the bottom most layer, the ordering in increasing value of the inner-products as said above induces the same ordering of the rows for each of these output matrices at the different ReLU gates. Similarly, the same ordering is induced on the columns (note that the orderings for the rows may be different from the ordering for the columns; what is important is that the rows have the same ordering across the family and similarly for the columns.)

Now we notice that the structure of the output matrices of the ReLU gates of the bottom most layer as described above is what is assumed in Claim 5.

Thus if $\{G_p(\mathbf{x}, \mathbf{y})\}_{p=1,..,w_1}$ are the output matrices at each of the ReLU gates in the bottom most layer, then for some $\mathbf{a} \in \mathbb{R}^{w_1}$ and $b \in \mathbb{R}$ at depth 2 the output matrix of any of the ReLU gates is given by, $\max\{0, bJ_{2^m \times 2^m} + \sum_{i=1}^{w_1} a_i G_i(\mathbf{x}, \mathbf{y})\}$ where $J_{2^m \times 2^m}$ is the matrix of all ones and the "max" is taken entrywise. Then the base case of the induction is settled by applying Claim 5 on this matrix, $bJ_{2^m \times 2^m} + \sum_{i=1}^{w_1} a_i G_i(\mathbf{x}, \mathbf{y})$ with $D = O(mW)$ and $w = w_1$.

We further note that the computations happening at the depth 2 ReLU gates obviously do not change

the ordering of the rows and columns frozen in at depth 1, i.e., in the G_i matrices. But with different depth 2 gates, i.e., different choices of the vectors $\mathbf{a}$ and the number b, because of the linearity (in $\mathbf{a}$ and b) of the operation of forming, $bJ_{2^m \times 2^m} + \sum_{i=1}^{w_1} a_i G_i(\mathbf{x}, \mathbf{y})$ they all have the same contiguous pattern of constant valued submatrices. Thus the depth 2 output matrices continue to satisfy the hypothesis of Claim 5.

To complete the induction step, we consider a family of ReLU circuits with depth $k+1$ corresponding to different choices of, $b \in \mathbb{R}$ and $\mathbf{a} \in \mathbb{R}^{w_k}$ which can be seen as computing $g(\mathbf{x}, \mathbf{y}) = \max\{0, b + \sum_{p=1}^{w_k} a_p g_p(\mathbf{x}, \mathbf{y})\}$ where $\{g_p(\mathbf{x}, \mathbf{y})\}_{p=1,..,w_k}$ is a family of ReLU circuits of depth k who by induction satisfy the lemma. Thus the corresponding output matrices of these depth $k+1$ circuits satisfy,

$$G(\mathbf{x}, \mathbf{y}) = \max\{0, bJ_{2^m \times 2^m} + \sum_{p=1}^{w_k} a_p G_p(\mathbf{x}, \mathbf{y})\}$$

where G_p is the matrix form of g_p. Thus, the induction hypothesis applied to depth k would then tell us that the rows and columns of each matrix G_p can be partitioned contiguously into $O\big((\prod_{i=1}^{k-1} w_i)(mW)\big)$ such that G_p is constant valued within each block. Then, by Claim 5, the rows and columns of the matrix $bJ_{2^m \times 2^m} + \sum_{p=1}^{w_k} a_p G_p(\mathbf{x}, \mathbf{y})$ can be partitioned into $O\big((\prod_{i=1}^{k} w_i)(mW)\big)$ contiguous blocks. Moreover, this ordering of the rows and colums does not vary across the different circuits in the family, because they all have the same weights in the bottom most layer. Hence, the same ordering works for all the circuits in the family.

□

Chapter 3

Provable Training of a ReLU gate

3.1 A review of provable neural training

In this chapter we will prove results about trainability of a ReLU gate under more general settings than hitherto known till date. To the best of our knowledge about the state-of-the-art in deep-learning both empirical and population risk minimization questions are typically solvable in either of the following two mutually exclusive scenarios : *Scenario* 1 *: Semi-Realizable Data* i.e the data comes as tuples $\mathbf{z} = (\mathbf{x}, \mathbf{y})$ with $\mathbf{y}$ being the noise corrupted output of a net (of known architecture) when given $\mathbf{x}$ as an input. And *Scenario* 2 *: Semi-Agnostic Data* i.e data comes as tuples $\mathbf{z} = (\mathbf{x}, \mathbf{y})$ with no obvious functional relationship between $\mathbf{x}$ and $\mathbf{y}$ but there could be geometrical or statistical assumptions about the $\mathbf{x}$ and $\mathbf{y}$.

We note that its not very interesting to work in the fully agnostic setting as in that case training even a single ReLU gate can be SPN-hard as shown in Goel et al., 2016 On the other hand the simplifications that happen for infinitely large networks have been discussed since Neal, 1996 and this theme has had a recent resurgence in works like Chizat and Bach, 2018; Jacot, Gabriel, and Hongler, 2018. Eventually this lead to an explosion of literature getting linear time training of various kinds of neural nets when their width is a high degree polynomial in training set size, inverse accuracy and inverse confidence parameters (a very *unrealistic* regime), (Lee et al., 2018; Wu, Du, and Ward, 2019; Du et al., 2018; Su and Yang, 2019; Kawaguchi and Huang, 2019; Huang and Yau, 2019; Allen-Zhu, Li, and Song, 2019a; Allen-Zhu, Li, and Liang, 2019; Allen-Zhu, Li, and Song, 2019b; Du and Lee, 2018; Zou et al., 2018a; Zou and Gu, 2019; Arora et al., 2019c; Arora et al., 2019b; Li et al., 2019; Arora et al., 2019a; Lee et al., 2018). The essential proximity of this regime to kernel methods have been thought of separately in works like Allen-Zhu and Li, 2019; Wei et al., 2019

Even in the wake of this progress, it remains unclear as to how any of this can help establish rigorous guarantees about "smaller" neural networks or more pertinently for constant size neural nets which is a regime closer to what is implemented in the real world. Thus motivated we can summarize what is open about training depth 2 nets into the following two questions,

1. **Question** 1 Can *any* algorithm train a ReLU gate to $\epsilon-$accuracy in poly$(\text{input}-\text{dimension}, \frac{1}{\epsilon})$ time using neither symmetry nor compact support assumptions on the distribution?

    - **Question** 1.5 Can a single ReLU gate be trained using (Stochastic) Gradient Descent with (a) random/arbitrary initialization and (b) weakly constrained data distribution - at least allowing it to be non-Gaussian and preferably non-compactly supported?

2. **Question** 2 Can a neural training algorithm work with the following naturally wanted properties being simultaneously true?

    (a) Nets of depth 2 with a constant/small number of gates.

    (b) The training data instances (and maybe also the noise) would have non-Gaussian non-compactly supported distributions.

    (c) Less structural assumptions on the weight matrices than being of the single filter convolutional type.

    (d) $\epsilon-$approximate answers be obtainable in at most poly$(\text{input}-\text{dimension}, \frac{1}{\epsilon})$ time.

3.2 A summary of our results

We make progress on some of the above fronts by drawing inspiration from two distinct streams of literature and often generalizing and blending techniques from them. First of them are the different avatars of the iterative stochastic non-gradient "Tron" algorithms analyzed in the past like, Rosenblatt, 1958; Pal and Mitra, 1992; Freund and Schapire, 1999; Kakade et al., 2011; Klivans and Meka, 2017; Goel and Klivans, 2017; Goel, Klivans, and Meka, 2018. The second kind of historical precedence that we are motivated by are the different works which have shown how some of the desired theorems about gradient descent can be proven if designed noise is injected into the algorithm in judicious ways, (Raginsky, Rakhlin, and Telgarsky, 2017; Xu et al., 2018; Zhang, Liang, and Charikar, 2017; Durmus and Majewski, 2019; Lee, Mangoubi, and Vishnoi, 2019; Jin et al., 2018; Mou et al., 2018; Li, Luo, and Qiao, 2019). Here we will be working with the simplest neural net which is just a single ReLU gate mapping $\mathbb{R}^n \ni \mathbf{x} \mapsto \max\{0, \mathbf{w}^\top \mathbf{x}\} \in \mathbb{R}$ for $\mathbf{w} \in \mathbb{R}^n$ being its weight. In here already the corresponding empirical or the population risk is neither convex nor smooth in how it depends on the weights. Thus to the best of our knowledge none of the convergence results among

these provable noise assisted algorithms cited above can be directly applied to this case because these proofs crucially leverage either convexity or very strong smoothness assumptions on the optimization objective. We show 3 kinds results in this chapter.

In Section 3.3 we have shown a very simple iterative stochastic algorithm to recover the underlying parameter $\mathbf{w}_*$ of the ReLU gate when realizable data allowed to be sampled online is of the form $(\mathbf{x}, \max\{0, \mathbf{w}_*^\top \mathbf{x}\})$. The distributional condition is very mild which essentially just captures the intuition that enough of our samples are such that $\mathbf{w}_*^\top \mathbf{x} > 0$.

Not only is our algorithm's run-time near-optimal, but to the best of our knowledge the previous attempts at this problem have solved this only for the Gaussian distribution (Soltanolkotabi, 2017; Kalan, Soltanolkotabi, and Avestimehr, 2019). Some results like Goel, Klivans, and Meka, 2018 included a solution to this above problem as a special case of their result while assuming that the the data distribution is having a p.d.f symmetric about the origin. Thus in contrast to all previous attempts our assumptions on the distribution are significantly milder.

In Section 3.4 we show the first-of-its-kind analysis of gradient descent on a ReLU gate albeit when assisted with the injection of certain kinds of noise. We assume that the labels in the data are realizable but we make no assumptions on the distribution of the domain. We make progress by showing that such a noise assisted GD in such a situation has a "diffusive" behaviour about the global minima i.e after T steps of the algorithm starting from *anywhere*, w.h.p *all* the steps of the algorithm have been within $\sqrt{T}$ distance of the global minima of the function. The key idea here is that of "coupling" which shows that from the iterates of noise injected gradient descent on the squared loss of a ReLU gate one can create a discrete bounded difference super-martingale.

Remark. We would like to emphasize to the reader that in such a distribution free regime as above, *no* algorithm is expected to provably train. Also note that the result is parametric in the magnitude of the added noise and hence one can make the algorithm be arbitrarily close to being a pure gradient descent.

In Section 3.5 we re-analyze a known algorithm called "GLM-Tron" under more general conditions than previously to show how well it can do (empirical) risk minimization on any Lipschitz gate with Lipschitz constant < 2 (in particular a ReLU gate) in the noisily realizable setting while no assumptions are being made on the distribution of the noise beyond their boundedness - hence the noise can

be "adversarial". We also point out how the result can be improved under some assumptions on the noise making it more benign. Note that in contrast to the training result in Section 3.3 which used a stochastic algorithm, here we are using full-batch iterative updates to gain these extra abilities to deal with more general gates, (adversarial) noise and essentially no distributional assumptions on training data.

3.3 Almost distribution free learning of a ReLU gate

If data, $\mathbf{x}$, is being sampled from a distribution $\mathcal{D}$ and the corresponding true labels are being generated from a ReLU gate as $\max\{0, \mathbf{w}_*^\top \mathbf{x}\}$ for some $\mathbf{w}_* \in \mathbb{R}^n$ unknown to us, then the question of learning this ReLU gate in this realizable setting is essentially the task of trying to solve the following optimization problem while having only sample access to $\mathcal{D}$, $\min_{\mathbf{w}\in\mathbb{R}^n} \mathbb{E}_{\mathbf{x}\sim\mathcal{D}}\left[\left(\max\{0, \mathbf{w}^\top \mathbf{x}\} - \max\{0, \mathbf{w}_*^\top \mathbf{x}\}\right)^2\right]$

In contrast to all previous work we show the following simple algorithm which solves this learning problem to arbitrarily good accuracy assuming only very mild conditions on $\mathcal{D}$. We leverage the simple intuition that if we can get to see enough labels $y = \max\{0, \mathbf{w}_*^\top \mathbf{x}\}$ where $y > 0$ then $\mathbf{w}^*$ is just the answer to the linear regression problem on those samples.

Algorithm 2 Modified SGD for a ReLU gate

1: **Input:** Sampling access to a distribution $\mathcal{D}$ on $\mathbb{R}^n$.
2: **Input:** Oracle access to the true labels when queried with some $\mathbf{x} \in \mathbb{R}^n$
3: **Input:** An arbitrarily chosen starting point of $\mathbf{w}_1 \in \mathbb{R}^n$ and a constant $\alpha < 0$
4: **for** $t = 1, \ldots$ **do**
5: Sample $\mathbf{x}_t \sim \mathcal{D}$ and query the oracle with it.
6: The oracle replies back with $y_t = \max\{0, \mathbf{w}_*^\top \mathbf{x}_t\}$
7: Form the gradient-proxy,
$$\mathbf{g}_t := \alpha \mathbf{1}_{y_t > 0}(y_t - \mathbf{w}_t^\top \mathbf{x}_t)\mathbf{x}_t$$
8: $\mathbf{w}_{t+1} := \mathbf{w}_t - \eta \mathbf{g}_t$
9: **end for**

Theorem 3.3.1. We assume that the the data distribution $\mathcal{D}$ is s.t $\mathbb{E}\left[\|\mathbf{x}\|^4\right]$ and the covariance matrix $\mathbb{E}\left[\mathbf{x}\mathbf{x}^\top\right]$ exist. Suppose $\mathbf{w}_*$ is s.t $\mathbb{E}\left[\mathbf{1}_{\mathbf{w}_*^\top \mathbf{x}>0}\mathbf{x}\mathbf{x}^\top\right]$ is positive definite. Then if Algorithm 2 is run with $\alpha < 0$ and $\eta = \frac{\lambda_{\min}\left(\mathbb{E}\left[\mathbf{1}_{\mathbf{w}_*^\top \mathbf{x}>0}\mathbf{x}\mathbf{x}^\top\right]\right)}{|\alpha|\mathbb{E}\left[\mathbf{1}_{\mathbf{w}_*^\top \mathbf{x}>0}\cdot\|\mathbf{x}\|^4\right]}$ starting from starting from $\mathbf{w}_1 \in \mathbb{R}^n$ then for $\mathrm{T} = O\left(\log \frac{\|\mathbf{w}_1 - \mathbf{w}_*\|^2}{\epsilon^2\delta}\right)$ we would have,

$$\mathbb{P}\left[\|\mathbf{w}_\mathrm{T} - \mathbf{w}_*\|^2 \leq \epsilon^2\right] \geq 1 - \delta$$

□

It's clear that $\|\mathbf{w}_T - \mathbf{w}_*\|^2 \leq \epsilon^2 \implies \mathbb{E}_{\mathbf{x}\sim\mathcal{D}}\left[\left(\max\{0, \mathbf{w}_T^\top \mathbf{x}\} - \max\{0, \mathbf{w}_*^\top \mathbf{x}\}\right)^2\right] \leq \epsilon^2 \mathbb{E}\left[\|\mathbf{x}\|^2\right]$ and hence Algorithm 2 is in effect approximately solving the risk minimization problem that we set out to solve. Also note that (a) the above convergence hold starting from arbitrary initialization $\mathbf{w}_1$, (b) the proof will establish along the way that the assumptions being made in the theorem are enough to ensure that the choice of η above is strictly positive and (c) for ease of interpretation we can just set $\alpha = -1$ in the above and observe how closely the choice of $\mathbf{g}_t$ in Algorithm 2 resembles the stochastic gradient that is commonly used and is known to have great empirical success.

Proof of Theorem 3.3.1. Let the training data sampled till the iterate t be $S_t = \{(x_1, y_1), \ldots, (x_t, y_t)\}$ From the algorithm we know that the weight vector update at t-th iteration is $\mathbf{w}_{t+1} = \mathbf{w}_t + \eta \mathbf{g}_t$. Thus,

$$\begin{aligned}\|\mathbf{w}_{t+1} - \mathbf{w}_*\|^2 &= \|\mathbf{w}_t - \eta \mathbf{g}_t - \mathbf{w}_*\|^2 \\ &= \|\mathbf{w}_t - \mathbf{w}_*\|^2 + \eta^2 \|\mathbf{g}_t\|^2 - 2\eta \langle \mathbf{w}_t - \mathbf{w}_*, \mathbf{g}_t \rangle \end{aligned} \tag{3.1}$$

We overload the notation to also denote by S_t the $\sigma-$algebra generated by the random variables $\mathbf{x}_1, ..., \mathbf{x}_t$. Conditioned on S_{t-1}, $\mathbf{w}_t$ is determined while $\mathbf{w}_{t+1}$ and $\mathbf{g}_t$ are random and dependent on the random choice of $\mathbf{x}_t$.

$$\begin{aligned}\mathbb{E}_{(\mathbf{x}_t, y_t)}\left[\|\mathbf{w}_{t+1} - \mathbf{w}_*\|^2 \mid S_{t-1}\right] &= \mathbb{E}_{(\mathbf{x}_t, y_t)}\left[\|\mathbf{w}_t - \mathbf{w}_*\|^2 \mid S_{t-1}\right] \\ &\quad + \underbrace{(-2\alpha\eta)\mathbb{E}_{(\mathbf{x}_t, y_t)}\left[\left\langle \mathbf{w}_t - \mathbf{w}_*, \mathbf{1}_{y_t > 0}\left(y_t - \mathbf{w}_t^\top \mathbf{x}_t\right)\mathbf{x}_t \right\rangle \mid S_{t-1}\right]}_{\text{Term 1}} \\ &\quad + \underbrace{\eta^2 \mathbb{E}_{(\mathbf{x}_t, y_t)}\left[\|\mathbf{g}_t\|^2 \mid S_{t-1}\right]}_{\text{Term 2}}\end{aligned} \tag{3.2}$$

Now we simplify the last two terms of the RHS above, starting from the rightmost,

$$\text{Term 2} = \mathbb{E}\left[\|\eta \mathbf{g}_t\|^2 \mid S_{t-1}\right] = \eta^2\alpha^2\mathbb{E}\left[\mathbf{1}_{y_t>0}(y_t - \mathbf{w}_t^\top \mathbf{x}_t)^2 \cdot \|\mathbf{x}_t\|^2 \mid S_{t-1}\right]$$
$$= \eta^2\alpha^2 \cdot \mathbb{E}\left[\mathbf{1}_{y_t>0}\big(\max\{0, \mathbf{w}_*^\top \mathbf{x}_t\} - \mathbf{w}_t^\top \mathbf{x}_t\big)^2 \cdot \|\mathbf{x}_t\|^2 \mid S_{t-1}\right]$$
$$\leq \eta^2\alpha^2\mathbb{E}\left[\mathbf{1}_{y_t>0}\|\mathbf{w}_* - \mathbf{w}_t\|^2 \cdot \|\mathbf{x}_t\|^4 \mid S_{t-1}\right]$$
$$\leq \eta^2\alpha^2\|\mathbf{w}_* - \mathbf{w}_t\|^2 \times \mathbb{E}\left[\mathbf{1}_{\mathbf{w}_*^\top \mathbf{x}_t>0} \cdot \|\mathbf{x}_t\|^4\right]$$

Note that in the above step the quantity, $\mathbb{E}\left[\mathbf{1}_{\mathbf{w}_*^\top \mathbf{x}>0} \cdot \|\mathbf{x}\|^4\right]$ is finite and it is easy to see why this is true given that $\forall\ \mathbf{w}_*, \mathbb{E}\left[\mathbf{1}_{\mathbf{w}_*^\top \mathbf{x}>0} \cdot \|\mathbf{x}\|^4\right] \leq \mathbb{E}\left[\|\mathbf{x}\|^4\right]$ and we recall that the quantity in this upperbound has been assumed to be finite in the hypothesis of the theorem.

Now we simplify Term 1 to get,

$$\text{Term1} = -2\eta\alpha\mathbb{E}\left[\mathbf{1}_{y_t>0}\Big(y_t - \mathbf{w}_t^\top \mathbf{x}_t\Big) \cdot (\mathbf{w}_t - \mathbf{w}_*)^\top \mathbf{x}_t \,\Big|\, S_{t-1}\right]$$
$$= -2\eta\alpha\mathbb{E}\left[\mathbf{1}_{y_t>0}\Big(\max\{0, \mathbf{w}_*^\top \mathbf{x}_t\} - \mathbf{w}_t^\top \mathbf{x}_t\Big) \times (\mathbf{w}_t - \mathbf{w}_*)^\top \mathbf{x}_t \,\Big|\, S_{t-1}\right]$$
$$\leq -2\eta\alpha\mathbb{E}\left[(\mathbf{w}_* - \mathbf{w}_t)^\top \mathbf{1}_{y_t>0}\mathbf{x}_t\mathbf{x}_t^\top(\mathbf{w}_t - \mathbf{w}_*) \,\Big|\, S_{t-1}\right]$$
$$\leq -2\eta|\alpha|\mathbb{E}\left[(\mathbf{w}_t - \mathbf{w}_*)^\top \mathbf{1}_{y_t>0}\mathbf{x}_t\mathbf{x}_t^\top(\mathbf{w}_t - \mathbf{w}_*) \,\Big|\, S_{t-1}\right]$$
$$\leq -2\eta|\alpha|\lambda_{\min}\Big(\mathbb{E}\left[\mathbf{1}_{\mathbf{w}_*^\top \mathbf{x}_t>0>0}\mathbf{x}_t\mathbf{x}_t^\top\right]\Big)\|\mathbf{w}_t - \mathbf{w}^*\|^2 \tag{3.3}$$

In the above step we invoked that the quantity $\mathbb{E}\left[\mathbf{1}_{\mathbf{w}_*^\top \mathbf{x}_t>0>0}\mathbf{x}_t\mathbf{x}_t^\top\right]$ exists and its easy to see why this is true given that $\forall\ \mathbf{w}_*, \mathbb{E}\left[\mathbf{1}_{\mathbf{w}_*^\top \mathbf{x}>0}\mathbf{x}\mathbf{x}^\top\right] \leq \mathbb{E}\left[\mathbf{x}\mathbf{x}^\top\right]$ and we recall that the covariance occurring of the distribution has been assumed to be finite in the hypothesis of the theorem.

We can combine both the upper bounds obtained above into the RHS of equation (3.2) to get,

$$\mathbb{E}_{(\mathbf{x}_t,y_t)}\Big[\|\mathbf{w}_{t+1}-\mathbf{w}_*\|^2 \mid S_{t-1}\Big] \\ \leq \left(1-2\eta|\alpha|\times\lambda_{\min}\Big(\mathbb{E}\Big[\mathbf{1}_{\mathbf{w}_*^\top\mathbf{x}_t>0}\mathbf{x}_t\mathbf{x}_t^\top\Big]\Big)+\eta^2\alpha^2\times\mathbb{E}\Big[\mathbf{1}_{\mathbf{w}_*^\top\mathbf{x}_t>0}\cdot\|\mathbf{x}_t\|^4\Big]\right)\|\mathbf{w}_t-\mathbf{w}_*\|^2 \tag{3.4}$$

We note that the two expectations on the RHS are properties of the distribution of the data $\mathbf{x}$ i.e $\mathcal{D}$ and $\mathbf{w}_*$ and we make the notation explicitly reflect that. $\mathbf{x}_t$ is a random variable that is independent of $\mathbf{w}_t$ since $\mathbf{x}_t$ is independent of $\mathbf{x}_1,\ldots,\mathbf{x}_{t-1}$. Hence by taking total expectation of the above we have,

$$\mathbb{E}\Big[\|\mathbf{w}_{t+1}-\mathbf{w}_*\|^2\Big] \leq \left(1-2\eta|\alpha|\lambda_{\min}\Big(\mathbb{E}\Big[\mathbf{1}_{\mathbf{w}_*^\top\mathbf{x}>0}\mathbf{x}\mathbf{x}^\top\Big]\Big)+\eta^2\alpha^2\times\mathbb{E}\Big[\mathbf{1}_{\mathbf{w}_*^\top\mathbf{x}>0}\cdot\|\mathbf{x}\|^4\Big]\right)\mathbb{E}\Big[\|\mathbf{w}_t-\mathbf{w}_*\|^2\Big] \tag{3.5}$$

Now we see that for $X_t := \mathbb{E}\Big[\|\mathbf{w}_t-\mathbf{w}_*\|^2\Big]$ the above is a recursion of the form given in Lemma 3.E.1 with $c_2=0$, $C=\|\mathbf{w}_1-\mathbf{w}_*\|^2$, $\eta'=\eta|\alpha|$, $b=2\lambda_{\min}\Big(\mathbb{E}\Big[\mathbf{1}_{\mathbf{w}_*^\top\mathbf{x}>0}\mathbf{x}\mathbf{x}^\top\Big]\Big)$ and $c_1=\mathbb{E}\Big[\mathbf{1}_{\mathbf{w}_*^\top\mathbf{x}>0}\cdot\|\mathbf{x}\|^4\Big]$

Now we note the following inequality,

$$\mathbb{E}\Big[\mathbf{1}_{\mathbf{w}_*^\top\mathbf{x}>0}\cdot\|\mathbf{x}\|^4\Big]=\mathbb{E}\Big[\mathbf{1}_{\mathbf{w}_*^\top\mathbf{x}>0}\cdot(\mathbf{x}^\top\mathbf{x})^2\Big]=\mathbb{E}\Big[\Big((\mathbf{1}^{\frac{1}{4}}_{\mathbf{w}_*^\top\mathbf{x}>0}\mathbf{x})^\top(\mathbf{1}^{\frac{1}{4}}_{\mathbf{w}_*^\top\mathbf{x}>0}\mathbf{x})\Big)^2\Big]=\mathbb{E}\Big[\Big(\operatorname{Tr}\Big((\mathbf{1}^{\frac{1}{4}}_{\mathbf{w}_*^\top\mathbf{x}>0}\mathbf{x})^\top(\mathbf{1}^{\frac{1}{4}}_{\mathbf{w}_*^\top\mathbf{x}>0}\mathbf{x})\Big)\Big)^2\Big] \\ =\mathbb{E}\Big[\Big(\operatorname{Tr}\Big(\mathbf{1}^{\frac{1}{2}}_{\mathbf{w}_*^\top\mathbf{x}>0}\mathbf{x}\mathbf{x}^\top\Big)\Big)^2\Big]$$

We note that the function $\mathbb{R}^{n\times n}\ni\mathbf{Y}\mapsto\operatorname{Tr}^2(\mathbf{Y})\in\mathbb{R}$ is convex and hence by Jensen's inequality we have,

$$\mathbb{E}\Big[\mathbf{1}_{\mathbf{w}_*^\top\mathbf{x}>0}\cdot\|\mathbf{x}\|^4\Big]\geq\overset{2}{\operatorname{Tr}}(\mathbb{E}\Big[\mathbf{1}^{\frac{1}{2}}_{\mathbf{w}_*^\top\mathbf{x}>0}\mathbf{x}\mathbf{x}^\top\Big])=\Big(\sum_{i=1}^n\lambda_i\Big(\mathbb{E}\Big[\mathbf{1}_{\mathbf{w}_*^\top\mathbf{x}>0}\mathbf{x}\mathbf{x}^\top\Big]\Big)\Big)^2\geq n^2\lambda^2_{\min}\Big(\mathbb{E}\Big[\mathbf{1}_{\mathbf{w}_*^\top\mathbf{x}>0}\mathbf{x}\mathbf{x}^\top\Big]\Big)$$

In the above λ_i indicates the i^{th} largest eigenvalue of the PSD matrix in its argument. And in particular the above inequality implies that $c_1>\frac{b^2}{4}$. Now we recall that the assumptions in the theorem which ensure that $b>0$ and hence now we have $\frac{b}{c_1}>0$ and hence the step-length prescribed in the theorem statement is strictly positive.

Thus by invoking the first case of Lemma 3.E.1 we have that for $\eta' = \eta|\alpha| = \frac{b}{2c_1}$ we have $\forall \epsilon' > 0$, $\mathbb{E}\Big[\|\mathbf{w}_T - \mathbf{w}_*\|^2\Big] \leq \epsilon'^2$ for $T = O\Big(\log \frac{\|\mathbf{w}_1 - \mathbf{w}_*\|^2}{\epsilon'^2}\Big)$

Thus given a $\epsilon > 0, \delta \in (0,1)$ we choose $\epsilon'^2 = \epsilon^2\delta$ and then by Markov inequality we have what we set out to prove,

$$\mathbb{P}\Big[\|\mathbf{w}_T - \mathbf{w}_*\|^2\Big] \leq \epsilon^2\Big] \geq 1 - \delta$$

□

3.4 Dynamics of noise assisted gradient descent on a single ReLU gate

As noted earlier it remains a significant challenge to prove the convergence of SGD or GD for a ReLU gate except for Gaussian data distributions. Towards this open question, we draw inspiration from ideas in Lee, Mangoubi, and Vishnoi, 2019 and we focus on analyzing a noise assisted version of gradient dynamics on a ReLU gate in the realizable case as given in Algorithm 3. In this setting we will see that we have some non-trivial control on the behaviour of the iterates despite making no distributional assumptions about the training data beyond realizability.

Algorithm 3 Noise Assisted Gradient Dynamics on a single ReLU gate (realizable data)

1: **Input:** We assume being given a step-length sequence $\{\eta_t\}_{t=1,2,\ldots}$ and $\{(\mathbf{x}_i, y_i)\}_{i=1,\ldots,S}$ tuples where $y_i = f_{\mathbf{w}_*}(\mathbf{x}_i)$ for some $\mathbf{w}_* \in \mathbb{R}^n$ where $f_{\mathbf{w}}$ is s.t

$$\mathbb{R}^n \ni \mathbf{x} \mapsto f_{\mathbf{w}}(\mathbf{x}) = \mathrm{ReLU}(\mathbf{w}^\top \mathbf{x}) = \max\{0, \mathbf{w}^\top \mathbf{x}\} \in \mathbb{R}$$

2: Start at $\mathbf{w}_0$
3: **for** $t = 0, \ldots$ **do**
4: Choice of Sub-Gradient $:= \mathbf{g}_t = -\frac{1}{S}\sum_{i=1}^{S} \mathbf{1}_{\mathbf{w}_t^\top \mathbf{x}_i \geq 0}\Big(y_i - f_{\mathbf{w}_t}(\mathbf{x}_i)\Big)\mathbf{x}_i$
5: $\mathbf{w}_{t+1} := \mathbf{w}_t - \eta_t(\mathbf{g}_t + \xi_{t,1}) + \sqrt{\eta_t}\xi_{t,2}$ ▷ $\xi_{t,1}$ is 0 mean bounded random variable
6: ▷ $\xi_{t,2}$ is a 0 mean random variable s.t $\mathbb{E}\Big[\|\xi_{t,2}\|^2\Big] < n$
7: end for

Note that in the above algorithm the indicator functions occurring in the definition of $\mathbf{g}_t$ are for the condition $\mathbf{w}_t^\top \mathbf{x}_i \geq 0$ for the $i^{th}-$data point. Whereas for the $\mathbf{g}_t$ used in Algorithm 2 in the previous section the indicator was for the condition $y_t > 0$ and hence dependent on $\mathbf{w}_*$ rather than $\mathbf{w}_t$.

Theorem 3.4.1. We analyze Algorithm 3 with constant step length $\eta_t = \eta$ Let $C := \max_{i=1,\ldots,S} \|\mathbf{x}_i\|$, $S_1 > 0$ be s.t $\forall t = 1, \ldots, \|\xi_{t,1}\| \leq S_1$ and $\{\xi_{t,2}\}_{t=1,\ldots}$ be mean 0, i.i.d as say ξ_2 s.t $\mathbb{E}\Big[\|\xi_{t,2}\|^2\Big] < n$, $\forall t = 1, \ldots$

Then for any $i_{\max} \in \mathbb{Z}^+$, $\lambda > 0$, $C_L \in (0, \sqrt{n})$, $0 < \eta < \frac{1}{C^2\sqrt{2i_{\max}}}$ and $r_*^2 \geq \lambda + \|\mathbf{w}_0 - \mathbf{w}_*\|^2 + i_{\max}\Big\{2\eta^2(C^4 r_*^2 + S_1^2) + \eta n\Big\}$ we have,

$$\mathbb{P}\Bigg[\exists i \in \{1, \ldots, i_{\max}\} \mid \|\mathbf{w}_i - \mathbf{w}_*\| > r_*\Bigg]$$
$$\leq i_{\max}\Bigg(\mathbb{P}\Big[\|\xi_2\| > C_L\Big] + \exp\Bigg\{-\frac{\lambda^2}{2i_{\max}} \times \frac{1}{2\sqrt{\eta}C_L\Big(r_* + \eta(C^2 r_* + S_1)\Big) + \eta\Big(2S_1 r_* + n + 2C^2 r_*^2 + 2\eta(C^4 r_*^2 + S_1^2)\Big)}\Bigg\}\Bigg)$$

Remark. Thus for η small enough and if $\mathbb{P}\Big[\|\xi_2\| > C_L\Big]$ is small then with significant probability the noise assisted gradient dynamics on a single ReLU gate in its first $i_{\max}$ steps remains confined inside a ball around the true parameter of radius,

$$r_* \geq \sqrt{\frac{\lambda + \|\mathbf{w}_0 - \mathbf{w}_*\|^2 + i_{\max}(\eta d + 2\eta^2 S_1^2)}{1 - 2i_{\max}\eta^2 C^4}}$$

Larger the $\lambda > 0$ we choose greater the (exponential) suppression in the probability that we get of finding the iterates outside the ball of radius r_* around the origin whereby r_* scales as $\sqrt{\lambda}$.

Also we note that for the above two natural choices of the distribution for $\{\xi_{t,2}, t = 1, \ldots\}$ are (a) $\{\xi_{t,2} = 0, t = 1, \ldots\}$ and (b) $\{(\xi_{t,2})_i \sim \mathcal{N}(0, \sigma_i), i = 1, \ldots, n, t = 1, \ldots\}$ where the $\{\sigma_i, i = 1, \ldots, n\}$ can be chosen as follows : corresponding to this choice of distributions we invoke Equation 3.5 from Ledoux and Talagrand, 2013 to note that $\mathbb{P}\Big[\|\xi_2\| > C_L\Big] \leq 4e^{-\frac{C_L^2}{8\times\mathbb{E}\Big[\|\xi_2\|^2\Big]}}$. Thus for the guarantee in the theorem to be non-trivial we need, $e^{-\frac{C_L^2}{8\times\mathbb{E}\Big[\|\xi_2\|^2\Big]}} < \frac{1}{4i_{\max}}$. Now note that $\mathbb{E}\Big[\|\xi_2\|^2\Big] = \sum_{i=1}^n \sigma_i^2$ and hence the above condition puts a smallness constraint on the variances of the Gaussian noise depending on how large an $i_{\max}$ we want, $\sum_{i=1}^n \sigma_i^2 < \frac{C_L^2}{8\log(4i_{\max})}$

Proof of Theorem 3.4.1.

For convenience we will use the notation, $\tilde{\mathbf{g}}_t := \mathbf{g}_t + \xi_{t,1}$. Suppose that at the t^{th} iterate we have that $\|\mathbf{w}_t - \mathbf{w}_*\| \leq r_*$. Given this we will get an upperbound on how far can $\mathbf{w}_{t+1}$ be from $\mathbf{w}_*$. Towards this we observe that,

$$\begin{aligned}\|\mathbf{w}_{t+1} - \mathbf{w}_*\|^2 &= \|\mathbf{w}_t - \eta_t\tilde{\mathbf{g}}_t + \sqrt{\eta_t}\xi_{t,2} - \mathbf{w}_*\|^2 \\ &= \|\mathbf{w}_t - \mathbf{w}_*\|^2 + \| - \eta_t\tilde{\mathbf{g}}_t + \sqrt{\eta_t}\xi_{t,2}\|^2 + 2\langle \mathbf{w}_t - \mathbf{w}_*, -\eta_t\tilde{\mathbf{g}}_t + \sqrt{\eta_t}\xi_{t,2}\rangle \end{aligned} \tag{3.6}$$

Expanding the second term above as, $\| - \eta_t\tilde{\mathbf{g}}_t + \sqrt{\eta_t}\xi_{t,2}\|^2 = \eta_t^2\|\tilde{\mathbf{g}}_t\|^2 + \eta_t\|\xi_{t,2}\|^2 - 2\eta_t^{3/2}\langle\tilde{\mathbf{g}}_t, \xi_{t,2}\rangle$

and combining into 3.6, we have,

$$\begin{aligned}&\|\mathbf{w}_{t+1} - \mathbf{w}_*\|^2 - \|\mathbf{w}_t - \mathbf{w}_*\|^2 \\ &= \langle\xi_{t,2}, -2\eta_t^{3/2}\tilde{\mathbf{g}}_t + 2\sqrt{\eta_t}\mathbf{w}_t - 2\sqrt{\eta_t}\mathbf{w}_*\rangle + \eta_t\|\xi_{t,2}\|^2 + \eta_t^2\|\tilde{\mathbf{g}}_t\|^2 - 2\eta_t\langle\mathbf{w}_t - \mathbf{w}_*, \tilde{\mathbf{g}}_t\rangle \\ &= \langle\xi_{t,2}, -2\eta_t^{3/2}\tilde{\mathbf{g}}_t + 2\sqrt{\eta_t}\mathbf{w}_t - 2\sqrt{\eta_t}\mathbf{w}_*\rangle + \eta_t\|\xi_{t,2}\|^2 + \eta_t^2\|\tilde{\mathbf{g}}_t\|^2 - 2\eta_t\langle\mathbf{w}_t - \mathbf{w}_*, \xi_{t,1}\rangle \\ &\quad - 2\eta_t\langle\mathbf{w}_t - \mathbf{w}_*, \mathbf{g}_t\rangle \\ &\leq -2\eta_t\langle\mathbf{w}_t - \mathbf{w}_*, \xi_{t,1}\rangle + \eta_t^2\|\tilde{\mathbf{g}}_t\|^2 + 2\sqrt{\eta_t}\langle\xi_{t,2}, -\eta_t\tilde{\mathbf{g}}_t + \mathbf{w}_t - \mathbf{w}_*\rangle + \eta_t\|\xi_{t,2}\|^2 \end{aligned} \tag{3.7}$$

In the last line we have used the Lemma 3.4.2 which shows this critical fact that $\langle\mathbf{w}_t - \mathbf{w}_*, \mathbf{g}_t\rangle \geq 0$.

Now we use the definition $\tilde{\mathbf{g}}_t = \mathbf{g}_t + \xi_{t,1}$ on the 2^{nd} term in the RHS of equation 3.7 to get,

$$\begin{aligned}&\|\mathbf{w}_{t+1} - \mathbf{w}_*\|^2 - \|\mathbf{w}_t - \mathbf{w}_*\|^2 \\ &\leq -2\eta_t\langle\mathbf{w}_t - \mathbf{w}_*, \xi_{t,1}\rangle + 2\eta_t^2(\|\mathbf{g}_t\|^2 + \|\xi_{t,1}\|^2) + 2\sqrt{\eta_t}\langle\xi_{t,2}, -\eta_t\tilde{\mathbf{g}}_t + \mathbf{w}_t - \mathbf{w}_*\rangle + \eta_t\|\xi_{t,2}\|^2 \\ &\leq -2\eta_t\langle\mathbf{w}_t - \mathbf{w}_*, \xi_{t,1}\rangle + 2\eta_t^2(\|\mathbf{g}_t\|^2 + S_1^2) - 2\sqrt{\eta_t}\langle\xi_{t,2}, \eta_t\tilde{\mathbf{g}}_t\rangle + 2\sqrt{\eta_t}\langle\xi_{t,2}, \mathbf{w}_t - \mathbf{w}_*\rangle + \eta_t\|\xi_{t,2}\|^2 \\ &\leq 2\eta_t^2(\|\mathbf{g}_t\|^2 + S_1^2) + \eta_t n + \Big[-2\eta_t\langle\mathbf{w}_t - \mathbf{w}_*, \xi_{t,1}\rangle - 2\sqrt{\eta_t}\langle\xi_{t,2}, \eta_t\tilde{\mathbf{g}}_t\rangle + 2\sqrt{\eta_t}\langle\xi_{t,2}, \mathbf{w}_t - \mathbf{w}_*\rangle + \eta_t(\|\xi_{t,2}\|^2 - n)\Big] \end{aligned} \tag{3.8}$$

Now we will get a finite bound on $\|\mathbf{g}_t\|^2$ by invoking the definition of r_* and C as follows,

$$\mathbf{g}_t = \frac{1}{S}\sum_{i=1}^{S}\Big(y_i - \mathrm{ReLU}(\mathbf{w}_t^\top \mathbf{x}_i)\Big)\mathbf{1}_{\mathbf{w}_t^\top \mathbf{x}_i \geq 0}(-\mathbf{x}_i) \implies \|\mathbf{g}_t\| \leq \frac{1}{S} \times C\sum_{i=1}^{S}|(\mathbf{w}_* - \mathbf{w}_t)^\top \mathbf{x}_i| \leq C^2 r_* \quad (3.9)$$

Substituting this back into equation 3.8 we have,

$$\begin{aligned}
&\|\mathbf{w}_{t+1} - \mathbf{w}_*\|^2 - \|\mathbf{w}_t - \mathbf{w}_*\|^2 \\
&\leq \Big\{2\eta_t^2(C^4 r_*^2 + S_1^2) + \eta_t n\Big\} \\
&+ \Big[-2\eta_t\langle \mathbf{w}_t - \mathbf{w}_*, \xi_{t,1}\rangle - 2\sqrt{\eta_t}\langle \xi_{t,2}, \eta_t\tilde{\mathbf{g}}_t\rangle + 2\sqrt{\eta_t}\langle \xi_{t,2}, \mathbf{w}_t - \mathbf{w}_*\rangle + \eta_t(\|\xi_{t,2}\|^2 - n)\Big]
\end{aligned} \quad (3.10)$$

We define $\mathbf{w}'_0 = \mathbf{w}_0$ and $\xi'_{t,2} = \min\Big\{C_L, \|\xi_{t,2}\|\Big\}\frac{\xi_{t,2}}{\|\xi_{t,2}\|}$ and $C_L \in (0, \sqrt{n})$.

Now we define a delayed stochastic process associated to the given algorithm,

$$\mathbf{w}'_{t+1} = \mathbf{w}'_t\mathbf{1}_{\|\mathbf{w}'_t - \mathbf{w}_*\| \geq r_*} + \Big(\mathbf{w}'_t - \eta_t\tilde{\mathbf{g}}_t + \sqrt{\eta_t}\xi'_{t,2}\Big)\mathbf{1}_{\|\mathbf{w}'_t - \mathbf{w}_*\| < r_*}$$

In the above we note that whenever the primed iterate steps out of the r_* ball it is made to stop. Associated to the above we define another stochastic process as follows,

$$z_t := \|\mathbf{w}'_t - \mathbf{w}_*\|^2 - t\Big\{2\eta_t^2(C^4 r_*^2 + S_1^2) + \eta_t n\Big\} \quad (3.11)$$

In Lemma 3.4.3 we prove the crucial property that for $\eta_t = \eta > 0$ a constant, the stochastic process $\{z_t\}_{t=0,1,\ldots}$ is a bounded difference process i.e $|z_{t+1} - z_t| \leq k$ for all $t = 1, \ldots$ and

$$k = 2\sqrt{\eta}C_L\Big(r_* + \eta(C^2 r_* + S_1)\Big) + \eta\Big(2S_1 r_* + n + 2C^2 r_*^2 + 2\eta(C^4 r_*^2 + S_1^2)\Big)$$

Now note that z_0 is a constant since $\mathbf{w}_0$ is so. The proof of Lemma 3.4.3 splits the analysis into two cases which we revisit again : in **Case** 1 in there we have $z_{t+1} - z_t < 0$ for $\eta_t = \eta > 0$. And in **Case** 2 therein we take a conditional expectation of the RHS of equation 3.16 w.r.t the sigma-algebra $\mathcal{F}_t$ generated by $\{z_0, \ldots, z_t\}$. Then the first two terms will go to 0 and the last term will give a negative

contribution since $\|\xi'_{t,2}\| \leq C_L$ and $C_L^2 < n$ by definition.

Thus the stochastic process $z_0, \ldots$ satisfies the conditions of the concentration of measure Theorem 3.D.1 and thus we get that for any $\lambda > 0$ and $t > 0$ and k as defined above,

$$\mathbb{P}\Big[z_t - z_0 \geq \lambda\Big] \leq e^{-\frac{\lambda^2}{2tk}}$$

And explicitly the above is equivalent to,

$$\begin{aligned}
&\mathbb{P}\Big[\|\mathbf{w}'_t - \mathbf{w}_*\|^2 - t\Big\{2\eta^2(C^4 r_*^2 + S_1^2) + \eta n\Big\} - \|\mathbf{w}_0 - \mathbf{w}_*\|^2 \geq \lambda\Big] \\
&\leq \exp\left\{-\frac{\lambda^2}{2t} \times \frac{1}{2\sqrt{\eta}C_L\Big(r_* + \eta(C^2 r_* + S_1)\Big) + \eta\Big(2S_1 r_* + n + 2C^2 r_*^2 + 2\eta(C^4 r_*^2 + S_1^2)\Big)}\right\}
\end{aligned} \tag{3.12}$$

The definition of r_* given in the theorem statement is that it satisfies, $r_*^2 \geq \lambda + \|\mathbf{w}_0 - \mathbf{w}_*\|^2 + t\Big\{2\eta^2(C^4 r_*^2 + S_1^2) + \eta n\Big\}$. Then the following is implied by equation 3.12,

$$\begin{aligned}
&\mathbb{P}\Big[\|\mathbf{w}'_t - \mathbf{w}_*\|^2 \geq r_*^2\Big] \\
&\leq \exp\left\{-\frac{\lambda^2}{2t} \times \frac{1}{2\sqrt{\eta}C_L\Big(r_* + \eta(C^2 r_* + S_1)\Big) + \eta\Big(2S_1 r_* + n + 2C^2 r_*^2 + 2\eta(C^4 r_*^2 + S_1^2)\Big)}\right\}
\end{aligned} \tag{3.13}$$

For the given positive integer $i_{\max}$ consider the event,

$$E := \Big\{\exists i \in \{1, \ldots, i_{\max}\} \mid \|\mathbf{w}_i - \mathbf{w}_*\| > r_*\Big\} \tag{3.14}$$

Define the event $E_t := \{\|\xi_{t,2}\| > C_L\}$. Thus, if E_t never happens, then the primed and the unprimed sequences both evolve the same unless $\mathbf{w}'_t$ leaves the r_* ball around $\mathbf{w}_*$ i.e., $\mathbf{w}'_t$ and $\mathbf{w}_t$ both leave the r_* ball around $\mathbf{w}_*$.

The sample space can be written as a disjoint union of events $A := \cup_{t=1}^{i_{\max}} E_t$ and $B := \cap_{t=1}^{i_{\max}} E_t^c$.

Let L_t be the event that t is the first time instant when $\mathbf{w}_t$ leaves the ball. Let L'_t be the event that t is the first time instant when $\mathbf{w}'_t$ leaves the ball. And we have argued above that when B happens

the two sequences evolve the same which in turn implies $B \cap L_t = B \cap L'_t$. Thus we have, $\mathbb{P}\Big[L_t\Big] = \mathbb{P}\Big[L_t \cap A\Big] + \mathbb{P}\Big[L_t \cap B\Big] = \mathbb{P}\Big[L_t \cap A\Big] + \mathbb{P}\Big[L'_t \cap B\Big]$

And combining this with E defined in 3.14 we have,

$$\mathbb{P}\Big[E\Big] = \sum_{t=1}^{i_{\max}} \mathbb{P}\Big[L_t\Big] = \sum_{t=1}^{i_{\max}} \Big(\mathbb{P}\Big[L_t \cap A\Big] + \mathbb{P}\Big[L'_t \cap B\Big]\Big) \leq \mathbb{P}\Big[A\Big] + \sum_{t=1}^{i_{\max}} \mathbb{P}\Big[L'_t\Big] \leq \sum_{t=1}^{i_{\max}} \Big(\mathbb{P}\Big[E_t\Big] + \mathbb{P}\Big[L'_t\Big]\Big)$$

The first equality and the first inequality above are true because L_t are disjoint events.

We further note that, $\mathbb{P}\Big[L'_t\Big] \leq \mathbb{P}\Big[\|\mathbf{w}'_t - \mathbf{w}_*\| > r_*\Big]$

Hence combining the above two inequalities we have,

$$\mathbb{P}\Big[\exists i \in \{1, \ldots, i_{\max}\} \mid \|\mathbf{w}_i - \mathbf{w}_*\| > r_*\Big] \leq \sum_{t=1}^{i_{\max}} \Big(\mathbb{P}\Big[\|\xi_{t,2}\| > C_L\Big] + \mathbb{P}\Big[\|\mathbf{w}'_t - \mathbf{w}_*\| > r_*\Big]\Big)$$

We invoke (a) the definition of the random variable ξ_2 and (b) equation 3.13 on each of the summands in the RHS above and we can infer that,

$$\mathbb{P}\Big[\exists i \in \{1, \ldots, i_{\max}\} \mid \|\mathbf{w}_i - \mathbf{w}_*\| > r_*\Big]$$
$$\leq i_{\max}\left(\mathbb{P}\Big[\|\xi_2\| > C_L\Big] + \exp\left\{-\frac{\lambda^2}{2i_{\max}} \times \frac{1}{2\sqrt{\eta}C_L\Big(r_* + \eta(C^2 r_* + S_1)\Big) + \eta\Big(2S_1 r_* + n + 2C^2 r_*^2 + 2\eta(C^4 r_*^2 + S_1^2)\Big)}\right\}\right)$$

This proves the theorem we wanted. □

Lemma 3.4.2. $\langle \mathbf{w}_t - \mathbf{w}_*, \mathbf{g}_t \rangle \geq 0$

Proof. We can obtain a (positive) lower bound on the inner product term $\langle \mathbf{w}_t - \mathbf{w}_*, \mathbf{g}_t \rangle$,

$$\begin{aligned}
&\langle \mathbf{w}_t - \mathbf{w}_*, \mathbf{g}_t \rangle \\
&= -\frac{1}{S}\sum_{i=1}^{S} \Big\langle \mathbf{w}_t - \mathbf{w}_*, \Big(y_i - \mathrm{ReLU}(\mathbf{w}_t^\top \mathbf{x}_i)\Big)\mathbf{1}(\mathbf{w}_t^\top \mathbf{x}_i \geq 0)\mathbf{x}_i \Big\rangle \\
&= -\frac{1}{S}\sum_{i=1}^{S} \Big(\mathbf{w}_t^\top \mathbf{x}_i - \mathbf{w}^{*\top}\mathbf{x}_i\Big)\Big(\mathrm{ReLU}(\mathbf{w}^{*\top}\mathbf{x}_i) - \mathrm{ReLU}(\mathbf{w}_t^\top \mathbf{x}_i)\Big)\mathbf{1}(\mathbf{w}_t^\top \mathbf{x}_i \geq 0) \\
&= \frac{1}{S}\sum_{i=1}^{S} \Big(\mathbf{w}^{*\top}\mathbf{x}_i - \mathbf{w}_t^\top \mathbf{x}_i\Big)\Big(\mathrm{ReLU}(\mathbf{w}^{*\top}\mathbf{x}_i) - \mathrm{ReLU}(\mathbf{w}_t^\top \mathbf{x}_i)\Big)\mathbf{1}(\mathbf{w}_t^\top \mathbf{x}_i \geq 0) \\
&\geq \frac{1}{S}\sum_{i=1}^{S} \Big(\mathrm{ReLU}(\mathbf{w}^{*\top}\mathbf{x}_i) - \mathrm{ReLU}(\mathbf{w}_t^\top \mathbf{x}_i)\Big)^2 \mathbf{1}(\mathbf{w}_t^\top \mathbf{x}_i \geq 0)
\end{aligned}$$

□ □

Lemma 3.4.3. If for all $t = 0, \ldots$ we have $\eta_t = \eta$ a constant > 0 then the stochastic process $\{z_t\}_{t=0,1,\ldots}$ defined in equation 3.11 is a bounded difference stochastic process i.e there exists a constant $k > 0$ s.t for all $t = 1, \ldots, |z_{t+1} - z_t| \leq k$

Proof. We have 2 cases to consider.

Case 1 : $\|\mathbf{w}_t' - \mathbf{w}_*\| \geq r_*$

$$\begin{aligned}
z_{t+1} - z_t &= -(t+1)(2\eta_{t+1}^2(C^4 r_*^2 + S_1^2) + \eta_{t+1} n) + t(2\eta_t^2(C^4 r_*^2 + S_1^2) + \eta_t n) \\
&= 2(C^4 r_*^2 + S_1^2)\Big\{t\eta_t^2 - (t+1)\eta_{t+1}^2\Big\} + n\Big\{t\eta_t - (t+1)\eta_{t+1}\Big\}
\end{aligned} \tag{3.15}$$

Case 2 : $\|\mathbf{w}_t' - \mathbf{w}_*\| < r_*$

Repeating the calculations as used to get equation 3.10 but with $\xi_{t,2}'$ instead of $\xi_{t,2}$ we will get,

$$z_{t+1} - z_t \leq -2\eta_t \langle \mathbf{w}_t - \mathbf{w}_*, \xi_{t,1} \rangle - 2\sqrt{\eta_t}\langle \xi_{t,2}', \eta_t \tilde{\mathbf{g}}_t \rangle + 2\sqrt{\eta_t}\langle \xi_{t,2}', \mathbf{w}_t - \mathbf{w}_* \rangle + \eta_t(\|\xi_{t,2}'\|^2 - n) \tag{3.16}$$

And by Cauchy-Schwartz the above implies,

$$z_{t+1} - z_t \leq 2\eta_t S_1 r_* + 2\sqrt{\eta_t}(r_* + \eta_t(C^2 r_* + S_1))C_L + \eta_t(C_L^2 - d) \tag{3.17}$$

Further repeating the calculations as used to get equation 3.10 but with $\xi'_{t,2}$ instead of $\xi_{t,2}$ we will get,

$$\|\mathbf{w}'_t - \mathbf{w}_*\|^2 - \|\mathbf{w}'_{t+1} - \mathbf{w}_*\|^2 \leq 2\eta_t r_*(\|\mathbf{g}_t\| + \|\xi_{t,1}\|) + 2\sqrt{\eta_t}\left\langle \xi'_{t,2}, -\eta_t \tilde{\mathbf{g}}_t + (\mathbf{w}_t - \mathbf{w}_*)\right\rangle$$

In the above we invoke the definition of S_1 and equation 3.9 to get,

$$\begin{aligned}\|\mathbf{w}'_t - \mathbf{w}_*\|^2 - \|\mathbf{w}'_{t+1} - \mathbf{w}_*\|^2 &\leq 2\eta_t r_*(C^2 r_* + S_1) + 2\sqrt{\eta_t}\left\langle \xi'_{t,2}, -\eta_t \tilde{\mathbf{g}}_t + (\mathbf{w}_t - \mathbf{w}_*)\right\rangle \\ &\leq 2\eta_t r_*(C^2 r_* + S_1) + 2\sqrt{\eta_t} C_L(\eta_t(C^2 r_* + S_1) + r_*)\end{aligned}$$

Hence we have,

$$z_t - z_{t+1} \leq 2\eta_t^2(C^4 r_*^2 + S_1^2) + \eta_t n + 2\eta_t r_*(C^2 r_* + S_1) + 2\sqrt{\eta_t} C_L(\eta_t(C^2 r_* + S_1) + r_*) \tag{3.18}$$

Combining equations 3.17 and 3.18 we have,

$$\begin{aligned}|z_t - z_{t+1}| \leq\ & 2\eta_t S_1 r_* + 2\sqrt{\eta_t} C_L\Big(r_* + \eta_t(C^2 r_* + S_1)\Big) \\ & + \max\left\{\eta_t(C_L^2 - n), \eta_t\Big[(n + 2C^2 r_*^2) + 2\eta_t(C^4 r_*^2 + S_1^2)\Big]\right\}\end{aligned} \tag{3.19}$$

If we now invoke the that $\eta_t = \eta$, a positive constant then the above and the previous equation 3.15 can be further combined to get for all $t = 0, \ldots,$

$$
\begin{aligned}
|z_t - z_{t+1}| \leq \max \Bigg\{ & n\eta + 2(C^4 r_*^2 + S_1^2)\eta^2 \\
& , 2\eta S_1 r_* + 2\sqrt{\eta} C_L \Big(r_* + \eta(C^2 r_* + S_1) \Big) + \max \Big\{ \eta(C_L^2 - n), \eta \Big[(n + 2C^2 r_*^2) + 2\eta(C^4 r_*^2 + S_1^2) \Big] \Big\} \Bigg\} \\
& \leq \max \Bigg\{ n\eta + 2(C^4 r_*^2 + S_1^2)\eta^2 \\
& , 2\sqrt{\eta} C_L \Big(r_* + \eta(C^2 r_* + S_1) \Big) + \eta \Big(2S_1 r_* + n + 2C^2 r_*^2 + 2\eta(C^4 r_*^2 + S_1^2) \Big) \Bigg\} \\
& \leq 2\sqrt{\eta} C_L \Big(r_* + \eta(C^2 r_* + S_1) \Big) + \eta \Big(2S_1 r_* + n + 2C^2 r^2 + 2\eta(C^4 r_*^2 + S_1^2) \Big) \qquad (3.20)
\end{aligned}
$$

In the second inequality above we are using invoking our assumption that $C_L^2 < n$. And this proves the boundedness of the stochastic process $\{z_t\}$ as we set out to prove and a candidate k is the RHS above. □

3.5 GLM-Tron converges on certain Lipschitz gates with no symmetry assumption on the data

Algorithm 4 GLM-Tron

1: **Input:** $\{(\mathbf{x}_i, y_i)\}_{i=1,\ldots,m}$ and an "activation function" $\sigma : \mathbb{R} \to \mathbb{R}$
2: $\mathbf{w}_1 = 0$
3: **for** $t = 1, \ldots$ **do**
4: $\quad \mathbf{w}_{t+1} := \mathbf{w}_t + \frac{1}{m} \sum_{i=1}^{m} \Big(y_i - \sigma(\langle \mathbf{w}_t, \mathbf{x}_i \rangle) \Big) \mathbf{x}_i$ ▷ Define $h_t(\mathbf{x}) := \sigma\Big(\langle \mathbf{w}_t, \mathbf{x} \rangle \Big)$
5: **end for**

First we state the following crucial lemma,

Lemma 3.5.1. Assume that for all $i = 1, \ldots, S$ $\|\mathbf{x}_i\| \leq 1$ and in Algorithm 4, σ is a $L-$Lipschitz non-decreasing function. Given any $\mathbf{w}$ and W s.t at iteration t, we have $\|\mathbf{w}_t - \mathbf{w}\| \leq W$, define $\eta > 0$ s.t $\| \frac{1}{S} \sum_{i=1}^{S} \Big(y_i - \sigma(\langle \mathbf{w}, \mathbf{x}_i \rangle) \Big) \mathbf{x}_i \| \leq \eta$. Then it follows that $\forall t = 1, 2, \ldots,$

$$
\|\mathbf{w}_{t+1} - \mathbf{w}\|^2 \leq \|\mathbf{w}_t - \mathbf{w}\|^2 - \Big(\frac{2}{L} - 1 \Big) \tilde{L}_S(h_t) + \Big(\eta^2 + 2\eta W(L+1) \Big)
$$

where we have defined, $\tilde{L}_S(h_t) := \frac{1}{S} \sum_{i=1}^{S} \Big(h_t(x_i) - \sigma(\langle \mathbf{w}, \mathbf{x}_i \rangle) \Big)^2 = \frac{1}{S} \sum_{i=1}^{S} \Big(\sigma(\langle \mathbf{w}_t, x_i \rangle) - \sigma(\langle \mathbf{w}, \mathbf{x}_i \rangle) \Big)^2$

The above algorithm was introduced in Kakade et al., 2011 for bounded activations. Here we show the applicability of that idea for more general activations and also while having adversarial attacks on the labels. We give the proof of the above Lemma in Appendix 3.A. Now we will see in the following theorem and its proof as to how the above Lemma leads to convergence of the "effective-ERM", $\tilde{L}_S$ by GLM-Tron on a single gate.

Theorem 3.5.2. [GLM-Tron (Algorithm 4) solves the effective-ERM on a ReLU gate upto noise bound with minimal distributional assumptions] Assume that for all $i = 1, \ldots, S$ $\|\mathbf{x}_i\| \leq 1$ and the label of the i^{th} data point y_i is generated as, $y_i = \sigma(\langle \mathbf{w}_*, \mathbf{x}_i \rangle) + \xi_i$ s.t $\forall i, |\xi_i| \leq \theta$ for some $\theta > 0$ and $\mathbf{w}_* \in \mathbb{R}^n$. If σ is a $L-$Lipschitz non-decreasing function for $L < 2$ then in at most $T = \frac{\|\mathbf{w}_*\|}{\epsilon}$ GLM-Tron steps we would attain parameter value $\mathbf{w}_T$ s.t,

$$\tilde{L}_S(h_T) = \frac{1}{S}\sum_{i=1}^{S}\Big(\sigma(\langle \mathbf{w}_T, x_i \rangle) - \sigma(\langle \mathbf{w}_*, \mathbf{x}_i \rangle)\Big)^2 < \frac{L}{2-L}\Big(\epsilon + (\theta^2 + 2\theta W(L+1))\Big)$$

□

Remark. *Firstly* Note that in the realizable setting i.e when $\theta = 0$, the above theorem is giving an upperbound on the number of steps needed to solve the ERM on say a ReLU gate to $O(\epsilon)$ accuracy. *Secondly* observe that the above theorem does not force any distributional assumption on the ξ_i beyond the assumption of its boundedness. Thus the noise could as well be chosen "adversarially" upto the constraint on its norm.

The above Theorem is proven in Appendix 3.B. If we make some assumptions on the noise being somewhat benign then we can get the following.

Theorem 3.5.3 (Performance guarantees on the GLM-Tron (Algorithm 4) in solving the ERM problem with data labels being output of a ReLU gate corrupted by benign noise). Assume that the noise random variables $\xi_i, i = 1, \ldots, S$ are identically distributed as a centered random variable say ξ. Then for $T = \frac{\|\mathbf{w}\|}{\epsilon}$, we have the following guarantee on the (true) empirical risk after T iterations of GLM-Tron (say $\tilde{L}_S(h_T)$),

$$\mathbb{E}_{\{(\mathbf{x}_i,\xi_i)\}_{i=1,\ldots S}}\Big[L_S(h_T)\Big] \leq \mathbb{E}_{\xi}[\xi^2] + \frac{L}{2-L}\Big(\epsilon + (\theta^2 + 2\theta W(L+1))\Big)$$

□

The above is proven in Appendix 3.C. Here we note a slight generalization of the above that can be easily read off from the above.

Corollary 3.5.4. Suppose that instead of assuming $\forall i = 1, \ldots, S \ |\xi_i| \leq \theta$ we instead assume that the joint distribution of $\{\xi_i\}_{i=1,\ldots,S}$ is s.t $\mathbb{P}\Big[|\xi_i| \leq \theta \ \forall i \in \{1, \ldots, S\}\Big] \geq 1 - \delta$ Then it would follow that the guarantee of the above Theorem 3.5.3 still holds but now with probability $1 - \delta$ over the noise distribution.

3.6 Conclusion

In this chapter we have initiated a number of directions of investigation towards understanding the trainability of finite sized nets while making minimal assumptions about the distribution of the data. A lot of open questions emanate from here which await answers. Of them we would like to particularly emphasize the issue of seeking a generalization of the results of Section 3.3 and Section 3.4 to single filter depth 2 nets as given in Definition 16 below, which in many ways can be said to be the next more complicated case to consider,

Definition 16 (Single Filter Neural Nets Of Depth 2). Given a set of k matrices $\mathrm{A}_i \in \mathbb{R}^{r \times n}$, a $\mathbf{w} \in \mathbb{R}^r$ and an activation function $\sigma : \mathbb{R} \to \mathbb{R}$ we call the following depth 2, width k neural net to be a "single filter neural net" defined by the matrices $\mathrm{A}_1, \ldots, \mathrm{A}_k$

$$\mathbb{R}^n \ni \mathbf{x} \mapsto f_{\mathbf{w}}(\mathbf{x}) = \frac{1}{k} \sum_{i=1}^{k} \sigma\Big(\mathbf{w}^\top \mathrm{A}_i \mathbf{x}\Big) \in \mathbb{R}$$

and where σ is the "Leaky-ReLU" which maps as, $\mathbb{R} \ni y \mapsto \sigma(y) = y\mathbf{1}_{y \geq 0} + \alpha y \mathbf{1}_{y < 0}$ for some $\alpha \geq 0$

Note that the above class of nets includes any single ReLU gate for $\alpha = 0, k = 1, \mathrm{A}_1 = I_{n \times n}$ and it also includes any depth 2 convolutional neural net with a single filter by setting the $\mathrm{A}_i's$ to be $0/1$ matrices such that each row has exactly one 1 and each column has at most one 1.

We would like to point out that towards this goal it would be interesting to settle a critical intermediate problem which is to know whether the sequence of random variables generated by noisy gradient descent on a ReLU gate as given in Algorithm 3 have distributional convergence and if they do then to find the corresponding rate.

Appendix To Chapter 3

3.A Proof of Lemma 3.5.1

Proof. We observe that,

$$\begin{aligned}\|\mathbf{w}_t - \mathbf{w}\|^2 - \|\mathbf{w}_{t+1} - \mathbf{w}\|^2 &= \|\mathbf{w}_t - \mathbf{w}\|^2 - \|\Big(\mathbf{w}_t + \frac{1}{S}\sum_{i=1}^{S}\Big(y_i - \sigma(\langle \mathbf{w}_t, \mathbf{x}_i\rangle)\Big)\mathbf{x}_i\Big) - \mathbf{w}\|^2 \\ &= -\frac{2}{S}\sum_{i=1}^{S}\Big\langle\Big(y_i - \sigma(\langle \mathbf{w}_t, \mathbf{x}_i\rangle)\Big)\mathbf{x}_i, \mathbf{w}_t - \mathbf{w}\Big\rangle - \|\frac{1}{S}\sum_{i=1}^{S}\Big(y_i - \sigma(\langle \mathbf{w}_t, \mathbf{x}_i\rangle)\Big)\mathbf{x}_i\|^2 \\ &= \frac{2}{S}\sum_{i=1}^{S}\Big(y_i - \sigma(\langle \mathbf{w}_t, \mathbf{x}_i\rangle)\Big)\Big(\langle \mathbf{w}, \mathbf{x}_i\rangle - \langle \mathbf{w}_t, \mathbf{x}_i\rangle\Big) - \|\frac{1}{S}\sum_{i=1}^{S}\Big(y_i - \sigma(\langle \mathbf{w}_t, \mathbf{x}_i\rangle)\Big)\mathbf{x}_i\|^2\end{aligned} \tag{3.21}$$

Analyzing the first term in the RHS above we get,

$$\begin{aligned}&\frac{2}{S}\sum_{i=1}^{S}\Big(y_i - \sigma(\langle \mathbf{w}_t, \mathbf{x}_i\rangle)\Big)\Big(\langle \mathbf{w}, \mathbf{x}_i\rangle - \langle \mathbf{w}_t, \mathbf{x}_i\rangle\Big) \\ &= \frac{2}{S}\sum_{i=1}^{S}\Big(y_i - \sigma(\langle \mathbf{w}, \mathbf{x}_i\rangle) + \sigma(\langle \mathbf{w}, \mathbf{x}_i\rangle) - \sigma(\langle \mathbf{w}_t, \mathbf{x}_i\rangle)\Big)\Big(\langle \mathbf{w}, \mathbf{x}_i\rangle - \langle \mathbf{w}_t, \mathbf{x}_i\rangle\Big) \\ &= \frac{2}{S}\sum_{i=1}^{S}\Big\langle\Big(y_i - \sigma(\langle \mathbf{w}, \mathbf{x}_i\rangle)\Big)\mathbf{x}_i, \mathbf{w} - \mathbf{w}_t\Big\rangle + \frac{2}{S}\sum_{i=1}^{S}\Big(\sigma(\langle \mathbf{w}, \mathbf{x}_i\rangle) - \sigma(\langle \mathbf{w}_t, \mathbf{x}_i\rangle)\Big)\Big(\langle \mathbf{x}_i, \mathbf{w}\rangle - \langle \mathbf{x}_i, \mathbf{w}_t\rangle\Big) \\ &\geq -2\eta W + \frac{2}{S}\sum_{i=1}^{S}\Big(\sigma(\langle \mathbf{w}, \mathbf{x}_i\rangle) - \sigma(\langle \mathbf{w}_t, \mathbf{x}_i\rangle)\Big)\Big(\langle \mathbf{x}_i, \mathbf{w}\rangle - \langle \mathbf{x}_i, \mathbf{w}_t\rangle\Big)\end{aligned}$$

In the first term above we have invoked the definition of η and W given in the Lemma. Further since we are given that σ is non-decreasing and $L-$Lipschitz, we have for the second term in the RHS above,

$$\tfrac{2}{S}\textstyle\sum_{i=1}^{S}\Big(\sigma(\langle \mathbf{w}, \mathbf{x}_i\rangle) - \sigma(\langle \mathbf{w}_t, \mathbf{x}_i\rangle)\Big)\Big(\langle \mathbf{x}_i, \mathbf{w}\rangle - \langle \mathbf{x}_i, \mathbf{w}_t\rangle\Big) \geq \tfrac{2}{SL}\textstyle\sum_{i=1}^{S}\Big(\sigma(\langle \mathbf{w}, \mathbf{x}_i\rangle) - \sigma(\langle \mathbf{w}_t, \mathbf{x}_i\rangle)\Big)^2 =: \tfrac{2}{L}\tilde{L}_S(h_t)$$

Thus together we have,

$$\frac{2}{S}\sum_{i=1}^{S}\Big(y_i - \sigma(\langle \mathbf{w}_t, \mathbf{x}_i\rangle)\Big)\Big(\langle \mathbf{w}, \mathbf{x}_i\rangle - \langle \mathbf{w}_t, \mathbf{x}_i\rangle\Big) \geq -2\eta W + \frac{2}{L}\tilde{L}_S(h_t) \tag{3.22}$$

Now we look at the second term in the RHS of equation 3.21 and that gives us,

$$\begin{aligned}
&\|\frac{1}{S}\sum_{i=1}^{S}\Big(y_i - \sigma(\langle \mathbf{w}_t, \mathbf{x}_i\rangle)\Big)\mathbf{x}_i\|^2 = \|\frac{1}{S}\sum_{i=1}^{S}\Big(y_i - \sigma(\langle \mathbf{w}, \mathbf{x}_i\rangle) + \sigma(\langle \mathbf{w}, \mathbf{x}_i\rangle) - \sigma(\langle \mathbf{w}_t, \mathbf{x}_i\rangle)\Big)\mathbf{x}_i\|^2 \\
&\leq \|\frac{1}{S}\sum_{i=1}^{S}\Big(y_i - \sigma(\langle \mathbf{w}, \mathbf{x}_i\rangle)\Big)\mathbf{x}_i\|^2 + 2\|\frac{1}{S}\sum_{i=1}^{S}\Big(y_i - \sigma(\langle \mathbf{w}, \mathbf{x}_i\rangle)\Big)\mathbf{x}_i\| \times \|\frac{1}{S}\sum_{i=1}^{S}\Big(\sigma(\langle \mathbf{w}, \mathbf{x}_i\rangle) - \sigma(\langle \mathbf{w}_t, \mathbf{x}_i\rangle)\Big)\mathbf{x}_i\| \\
&+ \|\frac{1}{S}\sum_{i=1}^{S}\Big(\sigma(\langle \mathbf{w}, \mathbf{x}_i\rangle) - \sigma(\langle \mathbf{w}_t, \mathbf{x}_i\rangle)\Big)\mathbf{x}_i\|^2 \\
&\leq \eta^2 + 2\eta\|\frac{1}{S}\sum_{i=1}^{S}\Big(\sigma(\langle \mathbf{w}, \mathbf{x}_i\rangle) - \sigma(\langle \mathbf{w}_t, \mathbf{x}_i\rangle)\Big)\mathbf{x}_i\| + \|\frac{1}{S}\sum_{i=1}^{S}\Big(\sigma(\langle \mathbf{w}, \mathbf{x}_i\rangle) - \sigma(\langle \mathbf{w}_t, \mathbf{x}_i\rangle)\Big)\mathbf{x}_i\|^2
\end{aligned} \tag{3.23}$$

Now by Jensen's inequality we have,

$$\|\tfrac{1}{S}\textstyle\sum_{i=1}^{S}\Big(\sigma(\langle \mathbf{w}, \mathbf{x}_i\rangle) - \sigma(\langle \mathbf{w}_t, \mathbf{x}_i\rangle)\Big)\mathbf{x}_i\|^2 \leq \tfrac{1}{S}\textstyle\sum_{i=1}^{S}\Big(\sigma(\langle \mathbf{w}, \mathbf{x}_i\rangle) - \sigma(\langle \mathbf{w}_t, \mathbf{x}_i\rangle)\Big)^2 = \tilde{L}_S(h_t)$$

And we have from the definition of L and W,

$$\|\frac{1}{S}\sum_{i=1}^{S}\Big(\sigma(\langle \mathbf{w}, \mathbf{x}_i\rangle) - \sigma(\langle \mathbf{w}_t, \mathbf{x}_i\rangle)\Big)\mathbf{x}_i\| \leq \frac{L}{S}\sum_{i=1}^{S}\|\mathbf{w} - \mathbf{w}_t\| \leq L \times W$$

Substituting the above two into the RHS of equation 3.23 we have,

$$\|\frac{1}{S}\sum_{i=1}^{S}\Big(y_i - \sigma(\langle \mathbf{w}_t, \mathbf{x}_i\rangle)\Big)\mathbf{x}_i\|^2 \leq \eta^2 + 2\eta LW + \tilde{L}_S(h_t) \tag{3.24}$$

Now we substitute equations 3.22 and 3.24 into equation 3.21 to get,

$$\|\mathbf{w}_t - \mathbf{w}\|^2 - \|\mathbf{w}_{t+1} - \mathbf{w}\|^2 \geq \Big(-2\eta W + \frac{2}{L}\tilde{L}_S(h_t)\Big) - (\eta^2 + 2\eta LW + \tilde{L}_S(h_t))$$

The above simplifies to the inequality we claimed in the lemma i.e,

$$\|\mathbf{w}_{t+1} - \mathbf{w}\|^2 \leq \|\mathbf{w}_t - \mathbf{w}\|^2 - \Big(\frac{2}{L} - 1\Big)\tilde{L}_S(h_t) + \Big(\eta^2 + 2\eta W(L+1)\Big)$$

□

3.B Proof of Theorem 3.5.2

Proof. The equation defining the labels in the data-set i.e $y_i = \sigma(\langle \mathbf{w}_*, \mathbf{x}_i \rangle) + \xi_i$ with $|\xi_i| \leq \theta$ along with our assumption that, $\|\mathbf{x}_i\| \leq 1$ implies that , $\|\frac{1}{S}\sum_{i=1}^{S}\Big(y_i - \sigma(\langle \mathbf{w}_*, \mathbf{x}_i \rangle)\Big)\mathbf{x}_i\| \leq \theta$. Thus we can invoke the above Lemma 3.5.1 between the t^{th} and the $t+1^{th}$ iterate with $\eta = \theta$ and W as defined there to get,

$$\|\mathbf{w}_{t+1} - \mathbf{w}_*\|^2 \leq \|\mathbf{w}_t - \mathbf{w}_*\|^2 - \left[\Big(\frac{2}{L} - 1\Big)\tilde{L}_S(h_t) - (\theta^2 + 2\theta W(L+1))\right]$$

If $\tilde{L}_S(h_t) \geq \frac{L}{2-L}\Big(\epsilon + (\theta^2 + 2\theta W(L+1))\Big)$ then, $\|\mathbf{w}_{t+1} - \mathbf{w}\|^2 \leq \|\mathbf{w}_t - \mathbf{w}\|^2 - \epsilon$. Thus if the above lowerbound on $\tilde{L}_s(h_t)$ holds in the t^{th} step then at the start of the $(t+1)^{th}$ step we still satisfy, $\|\mathbf{w}_{t+1} - \mathbf{w}\| \leq W$. Since the iterations start with $\mathbf{w}_1 = 0$, in the first step we can choose $W = \|\mathbf{w}_*\|$. Thus in at most $\frac{\|\mathbf{w}\|}{\epsilon}$ steps of the above kind we can have a decrease in distance of the iterate to $\mathbf{w}$.

Thus in at most $T = \frac{\|\mathbf{w}\|}{\epsilon}$ steps we have attained,

$$\tilde{L}_S(h_T) = \frac{1}{S}\sum_{i=1}^{S}\Big(\sigma(\langle \mathbf{w}_T, x_i \rangle) - \sigma(\langle \mathbf{w}_*, \mathbf{x}_i \rangle)\Big)^2 < \frac{L}{2-L}\Big(\epsilon + (\theta^2 + 2\theta W(L+1))\Big)$$

And that proves the theorem we wanted. □ □

3.C Proof of Theorem 3.5.3

Proof. Let the true empirical risk at the $T^{th}-$iterate be defined as,

$$L_S(h_T) = \frac{1}{S}\sum_{i=1}^{S}\Big(\sigma(\langle \mathbf{w}_T, x_i \rangle) - \sigma(\langle \mathbf{w}_*, \mathbf{x}_i \rangle) - \xi_i\Big)^2$$

Then it follows that,

$$\tilde{L}_S(h_T) - L_S(h_T) = \frac{1}{S}\sum_{i=1}^{S}\Big(\sigma(\langle \mathbf{w}_T, x_i\rangle) - \sigma(\langle \mathbf{w}_*, \mathbf{x}_i\rangle)\Big)^2 - \frac{1}{S}\sum_{i=1}^{S}\Big(\sigma(\langle \mathbf{w}_T, x_i\rangle) - \sigma(\langle \mathbf{w}_*, \mathbf{x}_i\rangle) - \xi_i\Big)^2$$
$$= \frac{1}{S}\sum_{i=1}^{S}\xi_i\Big(-\xi_i + 2\sigma(\langle \mathbf{w}_T, x_i\rangle) - 2\sigma(\langle \mathbf{w}_*, \mathbf{x}_i\rangle)\Big) = -\frac{1}{S}\sum_{i=1}^{S}\xi_i^2 + \frac{2}{S}\sum_{i=1}^{S}\xi_i\Big(\sigma(\langle \mathbf{w}_T, x_i\rangle) - \sigma(\langle \mathbf{w}_*, \mathbf{x}_i\rangle)\Big)$$

By the assumption of ξ_i being an unbiased noise the second term vanishes when we compute, $\mathbb{E}_{\{(\mathbf{x}_i,\xi_i)\}_{i=1,\dots S}}\Big[\tilde{L}_S(h_T) - L_S(h_T)\Big]$ Thus we are led to,

$$\mathbb{E}_{\{(\mathbf{x}_i,\xi_i)\}_{i=1,\dots S}}\Big[\tilde{L}_S(h_T) - L_S(h_T)\Big] = -\frac{1}{m}\mathbb{E}_{\{\xi_i\}_{i=1,\dots S}}\Big[\sum_{i=1}^{m}\xi_i^2\Big] = -\frac{1}{m}\sum_{i=1}^{m}\mathbb{E}_{\{\xi_i\}}\Big[\xi_i^2\Big] = -\mathbb{E}_{\xi}[\xi^2]$$

For $T = \frac{\|\mathbf{w}\|}{\epsilon}$, we invoke the upperbound on $\tilde{L}_S(h_T)$ from the previous theorem and we can combine it with the above to say,

$$\mathbb{E}_{\{(\mathbf{x}_i,\xi_i)\}_{i=1,\dots S}}\Big[L_S(h_T)\Big] \leq \mathbb{E}_{\xi}[\xi^2] + \frac{L}{2-L}\Big(\epsilon + (\theta^2 + 2\theta W(L+1))\Big)$$

This proves the theorem we wanted. □ □

3.D Reviewing a variant of the Azuma-Hoeffding Inequality

Theorem 3.D.1. Suppose we have a real valued discrete stochastic process given as, $\{X_0, X_i, \dots\}$ and the following properties hold,

- X_0 is a constant
- (The bounded difference property) $\forall i = 0, 1, \dots \exists c_i > 0$ s.t $|X_i - X_{i-1}| \leq c_i$
- (The super-martingale property) $\forall i = 0, 1, \dots, \mathbb{E}\Big[X_i - X_{i-1} \mid \mathscr{F}_{i-1}\Big] \leq 0$ with $\mathscr{F}_{i-1} = \sigma\Big(\Big\{X_0, \dots, X_{i-1}\Big\}\Big)$

Then for any $\lambda > 0$ and a positive integer n we have the following concentration inequality,

$$\mathbb{P}\Big[X_n - X_0 \geq \lambda\Big] \leq e^{-\frac{1}{2}\frac{\lambda^2}{\sum_{i=1}^{n} c_i^2}}$$

Proof. We note that for any $c, t > 0$, the function $f(x) = e^{tx}$ lies below the straight line connecting the two points $(-c, f(-c))$ and $(c, f(c))$. This gives the inequality, $e^{tx} \leq e^{-tc} + \Big(\frac{e^{tc} - e^{-tc}}{2c}\Big)(x + c)$. This simplifies to,

$$e^{tx} \leq \frac{1}{2c}(e^{tc} - e^{-tc})x + \left(\frac{e^{tc} + e^{-tc}}{2}\right) \tag{3.25}$$

Note that the above inequality holds only when $|x| \leq c$ Now we can invoke the bounded difference property of $|X_i - X_{i-1}| \leq c_i$ and use equation 3.25 with $x = X_i - X_{i-1}$ and $c = c_i$ to get,

$$\mathbb{E}\left[e^{t(X_i - X_{i-1})} \mid \mathcal{F}_{i-1}\right] \leq \mathbb{E}\left[\frac{e^{tc_i} - e^{-tc_i}}{2c_i}\left(X_i - X_{i-1}\right) + \left(\frac{e^{tc_i} + e^{-tc_i}}{2}\right) \mid \mathcal{F}_{i-1}\right] \leq \frac{e^{tc_i} + e^{-tc_i}}{2}$$

The last inequality follows from the given property that, $\mathbb{E}\left[X_i - X_{i-1} \mid \mathcal{F}_{i-1}\right] \leq 0$

Now we invoke the inequality $\frac{e^x + e^{-x}}{2} \leq e^{\frac{x^2}{2}}$ on the RHS above to get,

$$\mathbb{E}\left[e^{t(X_i - X_{i-1})} \mid \mathcal{F}_{i-1}\right] \leq e^{\frac{t^2 c_i^2}{2}}$$

Further since X_{i-1} is $\mathcal{F}_{i-1}$ measurable we can write the above as, $\mathbb{E}\left[e^{tX_i} \mid \mathcal{F}_{i-1}\right] \leq e^{tX_{i-1}} e^{\frac{t^2 c_i^2}{2}}$

Now we recurse the above as follows,

$$\mathbb{E}\left[e^{tX_n}\right] = \mathbb{E}\left[\mathbb{E}\left[e^{tX_n} \mid \mathcal{F}_{n-1}\right]\right] \leq \mathbb{E}\left[e^{\frac{t^2 c_n^2}{2}} e^{tX_{n-1}}\right] = e^{\frac{t^2 c_n^2}{2}} \mathbb{E}\left[e^{tX_{n-1}}\right] \ldots \leq \prod_{i=1}^{n} e^{\frac{t^2 c_i^2}{2}} \mathbb{E}[e^{tX_0}]$$

Now invoking that X_0 is a constant we can rewrite the above as, $\mathbb{E}\left[e^{t(X_n - X_0)}\right] \leq e^{\frac{t^2}{2}\sum_{i=1}^{n} c_i^2}$

Hence for any $\lambda > 0$ we have by invoking the above,

$$\mathbb{P}\left[X_n - X_0 \geq \lambda\right] = \mathbb{P}\left[e^{t(X_n - X_0)} \geq e^{t\lambda}\right] \leq e^{-t\lambda}\mathbb{E}\left[e^{t(X_n - X_0)}\right] \leq e^{-t\lambda} e^{\frac{t^2}{2}\sum_{i=1}^{n} c_i^2}$$

Now choose $t = \frac{\lambda}{\sum_{i=1}^{n} c_i^2}$ and we get, $\mathbb{P}\left[X_n - X_0 \geq \lambda\right] \leq e^{-\frac{1}{2}\frac{\lambda^2}{\sum_{i=1}^{n} c_i^2}}$

□

3.E A recursion estimate

Lemma 3.E.1. Given constants $\eta', b, c_1, c_2 > 0$ suppose one has a sequence of real numbers $X_1 = C, X_2, ..$ s.t,

$$X_{t+1} \leq (1 - \eta' b + \eta'^2 c_1) X_t + \eta'^2 c_2$$

Given any $\epsilon' > 0$ in the following two cases we have, $X_{\mathrm{T}} \leq \epsilon'^2$

- If $c_2 = 0, c_1 > \frac{b^2}{4}, C > 0, \delta > 0$,
  $\eta' = \frac{b}{2c_1}$ and $\mathrm{T} = O\Big(\log \frac{C}{\epsilon'^2}\Big)$
- If $0 < c_2 \leq c_1, \epsilon'^2 \leq C, \frac{b^2}{c_1} \leq \left(\sqrt{\epsilon'} + \frac{1}{\sqrt{\epsilon'}}\right)^2$,
  $\eta' = \frac{b}{c_1} \cdot \frac{\epsilon'^2}{(1+\epsilon'^2)}$ and $\mathrm{T} = O\left(\frac{\log\left(\frac{\epsilon'^2(c_1-c_2)}{Cc_1 - c_2\epsilon'^2}\right)}{\log\left(1 - \frac{b^2}{c_1} \cdot \frac{\epsilon'^2}{(1+\epsilon'^2)^2}\right)}\right)$.

Proof. Suppose we define $\alpha = 1 - \eta' b + \eta'^2 c_1$ and $\beta = \eta'^2 c_2$. Then we have by unrolling the recursion,

$$X_t \leq \alpha X_{t-1} + \beta \leq \alpha(\alpha X_{t-1} + \beta) + \beta \leq ... \leq \alpha^{t-1} X_1 + \beta \frac{1 - \alpha^{t-1}}{1 - \alpha}.$$

We recall that $X_1 = C$ to realize that our Lemma gets proven if we can find T s.t,

$$\alpha^{\mathrm{T}-1} C + \beta \frac{1 - \alpha^{\mathrm{T}-1}}{1 - \alpha} = \epsilon'^2$$

Thus we need to solve the following for T s.t, $\alpha^{\mathrm{T}-1} = \frac{\epsilon'^2(1-\alpha) - \beta}{C(1-\alpha) - \beta}$

Case 1 : $\beta = 0$ In this case we see that if $\eta > 0$ is s.t $\alpha \in (0, 1)$ then $\alpha^{\mathrm{T}-1} = \frac{\epsilon'^2}{C} \implies \mathrm{T} = 1 + \frac{\log \frac{C}{\epsilon'^2 \delta}}{\log \frac{1}{\alpha}}$

But $\alpha = \eta'^2 c_1 - \eta' b + 1 = \left(\eta' \sqrt{c_1} - \frac{b}{2\sqrt{c_1}}\right)^2 + \left(1 - \frac{b^2}{4c_1}\right)$ Thus $\alpha \in (0, 1)$ is easily ensured by choosing $\eta' = \frac{b}{2c_1}$ and ensuring $c_1 > \frac{b^2}{4}$. This gives us the first part of the theorem.

Case 2 : $\beta > 0$

This time we are solving,

$$\alpha^{\mathrm{T}-1} = \frac{\epsilon'^2(1-\alpha)-\beta}{C(1-\alpha)-\beta} \tag{3.26}$$

Towards showing convergence, we want to set η' such that $\alpha^{t-1} \in (0,1)$ for all t. Since $\epsilon'^2 < C$, it is sufficient to require,

$$\beta < \epsilon'^2\delta(1-\alpha) \implies \alpha < 1 - \frac{\beta}{\epsilon'^2} \Leftrightarrow 1 - \frac{b^2}{4c_1} + \left(\eta'\sqrt{c_1} - \frac{b}{2\sqrt{c_1}}\right)^2 \leq 1 - \frac{\beta}{\epsilon'^2}$$
$$\Leftrightarrow \frac{\eta'^2 c_2}{\epsilon'^2} \leq \frac{b^2}{4c_1} - \left(\eta'\sqrt{c_1} - \frac{b}{2\sqrt{c_1}}\right)^2 \Leftrightarrow \frac{c_2}{\epsilon'^2} \leq \frac{b^2}{4c_1\eta'^2} - \left(\sqrt{c_1} - \frac{b}{2\sqrt{c_1}\eta'}\right)^2$$

Set $\eta' = \frac{b}{\theta c_1}$ for some constant $\theta > 0$ to be chosen such that,

$$\frac{c_2}{\epsilon'^2} \leq \frac{b^2}{4c_1 \cdot \frac{b^2}{\theta^2 c_1^2}} - \left(\sqrt{c_1} - \frac{b}{2\sqrt{c_1}\cdot\frac{b}{\theta c_1}}\right)^2 \implies \frac{c_2}{\epsilon'^2} \leq c_1\frac{\theta^2}{4} - c_1\cdot\left(\frac{\theta}{2}-1\right)^2 \implies c_2 \leq \epsilon'^2\cdot c_1(\theta-1)$$

Since $c_2 \leq c_1$ we can choose, $\theta = 1 + \frac{1}{\epsilon'^2}$ and we have $\alpha^{t-1} < 1$. Also note that,

$$\alpha = 1 + \eta'^2 c_1 - \eta' b = 1 + \frac{b^2}{\theta^2 c_1^2} - \frac{b^2}{\theta c_1} = 1 - \frac{b^2}{c_1}\cdot\left(\frac{1}{\theta} - \frac{1}{\theta^2}\right).$$
$$= 1 - \frac{b^2}{c_1}\cdot\frac{\epsilon'^2}{(1+\epsilon'^2)^2} = 1 - \frac{b^2}{c_1}\cdot\frac{1}{\left(\epsilon' + \frac{1}{\epsilon'}\right)^2}$$

And here we recall that the condition that the lemma specifies on the ratio $\frac{b^2}{c_1}$ which ensures that the above equation leads to $\alpha > 0$

Now in this case we get the given bound on T in the Lemma by solving equation 3.26. To see this, note that,

$$\alpha = 1 - \frac{b^2}{c_1}\cdot\frac{\epsilon'^2}{(1+\epsilon'^2)^2} \text{ and } \beta = \eta'^2 c_2 = \frac{b^2}{\theta^2 c_1}\cdot c_2 = \frac{b^2 c_2}{c_1}\cdot\frac{(\epsilon'^2)^2}{(1+\epsilon'^2)^2}.$$

Plugging the above into equation 3.26 we get, $\alpha^{\mathrm{T}-1} = \frac{\epsilon'^2\delta(c_1-c_2)}{Cc_1 - c_2\epsilon'^2} \implies \mathrm{T} = 1 + \frac{\log\left(\frac{\epsilon'^2(c_1-c_2)}{Cc_1 - c_2\epsilon'^2}\right)}{\log\left(1 - \frac{b^2}{c_1}\cdot\frac{\epsilon'^2}{(1+\epsilon'^2)^2}\right)}$ □

Chapter 4

Sparse Coding and Autoencoders

4.1 Introduction

One of the fundamental themes in learning theory is to consider data being sampled from a generative model and to provide efficient methods to recover the original model parameters exactly or with tight approximation guarantees. Classic examples include learning a mixture of gaussians (Moitra and Valiant, 2010), certain graphical models (Anandkumar et al., 2014), full rank square dictionaries (Spielman, Wang, and Wright, 2012; Błasiok and Nelson, 2016) and overcomplete dictionaries (Agarwal et al., 2014; Arora et al., 2014; Arora et al., 2015; Arora, Ge, and Moitra, 2014) The problem is usually distilled down to a non-convex optimization problem whose solution can be used to obtain the model parameters. With these hard non-convex problems it has been difficult to find any universal view as to why sometimes gradient descent gives very good and sometimes even exact recovery. In recent times progress has been made towards achieving a geometric understanding of the landscape of such non-convex optimization problems (Ge, Jin, and Zheng, 2017), (Mei, Bai, and Montanari, 2016), (Wu, Zhu, et al., 2017). The corresponding question of parameter recovery for neural nets with one layer of activation has been solved in some special cases, (Du, Lee, and Tian, 2017; Allen-Zhu, 2017; Janzamin, Sedghi, and Anandkumar, 2015; Sedghi and Anandkumar, 2014; Li and Yuan, 2017; Tian, 2017; Zhang et al., 2017). Almost all of these cases are in the supervised setting where it has also been assumed that the labels are being generated from a net of the same architecture as is being trained. In contrast to these works we address an unsupervised learning problem, and possibly more realistically, we do not tie the data generation model (sensing of sparse vectors by an overcomplete incoherent dictionary) to the neural architecture being analyzed except for assuming knowledge of a few parameters about the ground truth.

Here we specialize to the generative model of *dictionary learning/sparse coding* where one receives samples of vectors $y \in \mathbb{R}^n$ that have been generated as $y = A^* x^*$ where $A^* \in \mathbb{R}^{n \times h}$ and $x^* \in \mathbb{R}^h$.

We typically assume that the number of non-zero entries in x^* to be no larger than some function of the dimension h and that A^* satisfies certain incoherence properties. The question now is to recover A^* from samples of y. There have been renewed investigations into the hardness of this problem (Tillmann, 2015) and many former results have recently been reviewed in these lectures "CBMS Conference on Sparse Approximation and Signal Recovery Algorithms, May 22-26, 2017 and 16th New Mexico Analysis Seminar, May 21". This question has been a cornerstone of learning theory ever since the ground-breaking paper by Olshausen and Field (Olshausen and Field, 1997) (a recent review by the same authors can be found in Olshausen and Field, 2005). Over the years many algorithms have been developed to solve this problem and a detailed comparison among these various approaches can be found in Błasiok and Nelson, 2016.

Autoencoder neural networks that map $\mathbb{R}^n \to \mathbb{R}^n$ were defined in Section 1.1.1. These networks have been used extensively (Baldi, 2012; Bengio et al., 2013; Rifai et al., 2011; Vincent et al., 2008; Vincent et al., 2010) in the past for unsupervised feature learning tasks, and have been found to be successful in generating discriminative features (Coates, Ng, and Lee, 2011). A number of different autoencoder architectures and regularizers have been proposed which purportedly induce sparsity, at the hidden layer (Arpit et al., 2016; Coates and Ng, 2011; Li et al., 2016; Ng, 2011). There has also been some investigation into what autoencoders learn about the data distribution (Alain and Bengio, 2014).

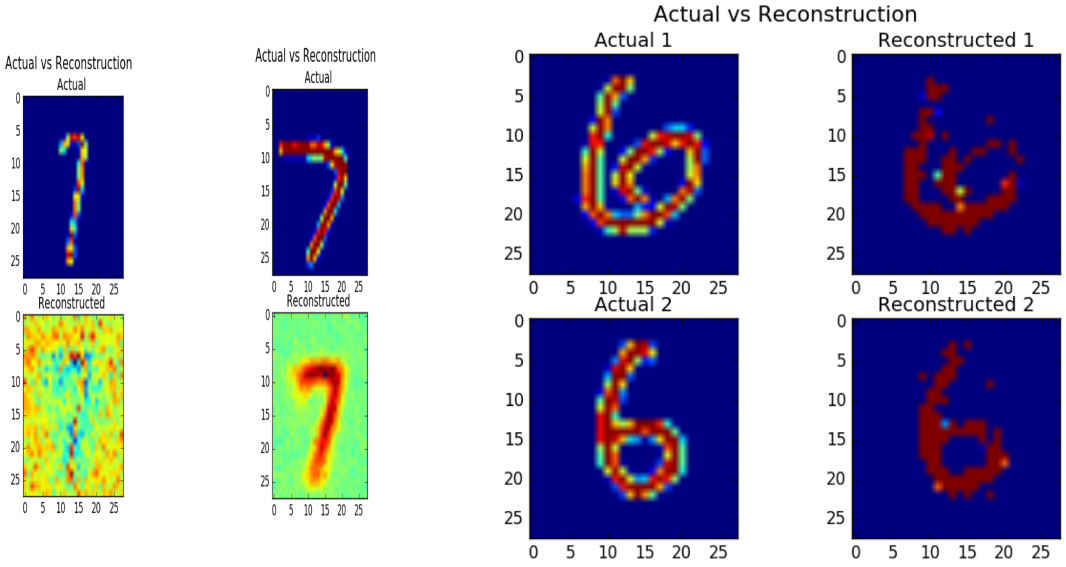
Olshausen and Field had, as early as 1996, already made the connection between sparse coding and training neural architectures and in today's terminology this problem is very naturally reminiscent of the architecture of an autoencoder (Olshausen and Field, 1996). However, to the best of our knowledge, there has not been sufficient progress to rigorously establish whether autoencoders can do sparse coding.

In this work, we present our progress towards bridging the above mentioned mathematical gap. To the best of our knowledge, there is no theoretical evidence (even under the usual generative assumptions of sparse coding) that the stationary points of any of the usual squared loss functions (with or without any of the usual regularizers) have any resemblance to the original dictionary that is being sought to be learned. **The main point of this paper is to rigorously prove that for autoencoders with ReLU activation, the standard squared loss function has a neighborhood around the dictionary A^* where the norm of the expected gradient is very small (for large enough sparse code dimension h). Thus, all points in a neighborhood of A^*, including A^*, are all asymptotic critical points of this standard squared loss.** We supplement our theoretical result with experimental evidence for it

in Section 4.6, which also strongly suggests that the standard squared loss function has a local minimum in a neighborhood around A^*. We believe that our results provide theoretical and experimental evidence that the sparse coding problem can be tackled by training autoencoders.

4.1.1 A motivating experiment on MNIST using TensorFlow

We used TensorFlow (Abadi et al., 2016) to train two ReLU autoencoders mapping $\mathbb{R}^{784} \to \mathbb{R}^{784}$ (since the MNIST images vectorize to elements in $\mathbb{R}^{784}$). These networks were trained on a subset of the MNIST dataset of handwritten digits. One of the nets had a single hidden layer of size 10000 and the other one had two hidden layers of size 5000 and 784 (and a fixed identity matrix giving the output from the second layer of activations). In both the cases the weights of the encoder and decoder were maintained as transposes of each other. We trained the autoencoders on the standard squared loss function using RMSProp "RMSprop Gradient Optimization". The training was done on 6000 images of the digits 6 and 7 from the MNIST dataset. In the following panel we show four pairs (two for each net) of "reconstructed" image i.e output of the trained net when its given as input the "actual" photograph as input.



1 layer Relu Autoencoder

2 layer Relu Autoencoder

In our opinion, the above figures add support to the belief that a single and a double layer ReLU activated $\mathbb{R}^n \to \mathbb{R}^n$ network can learn an implicit high dimensional structure about the handwritten digits dataset. In particular this demonstrates that though adding more hidden layers obviously helps enhance the reconstruction ability, the single hidden layer autoencoder do hold within them significant power for unsupervised learning of representations. Unfortunately analyzing the RMSProp update rule used in the above experiment seems to be currently beyond our analytic means - though in the next chapter we shall make some progress about understanding this algorithm. However, we take inspiration from these experiments to devise a different mathematical set-up which is much more amenable to analysis taking us towards a better understanding of the power of autoencoders.

4.2 Introducing the neural architecture and the distributional assumptions

For the autoencoders we continue to use the same variables as defined in equation 1.3.

Assumptions on the dictionary and the sparse code. We assume that our signal y is generated using sparse linear combinations of atoms/vectors of an overcomplete dictionary, i.e., $y = A^* x^*$, where $A^* \in \mathbb{R}^{n \times h}$ is a dictionary, and $x^* \in (\mathbb{R}^{\geq 0})^h$ is a non-negative sparse vector, with at most $k = h^p$ (for some $0 < p < 1$) non zero elements. The columns of the original dictionary A^* (labeled as $\{A_i^*\}_{i=1}^h$) are assumed to be normalized and we parameterize its incoherence property as, $\max_{\substack{i,j=1,..,h \\ i \neq j}} |\langle A_i^*, A_j^* \rangle| \leq \frac{\mu}{\sqrt{n}} = h^{-\xi}$ for some $\xi > 0$.

We assume that the sparse code x^* is sampled from a distribution with the following properties. We fix a set of possible supports of x^*, denoted by $\mathcal{S} \subseteq 2^{[h]}$, where each element of $\mathcal{S}$ has at most $k = h^p$ elements. We consider any arbitrary discrete probability distribution $D_{\mathcal{S}}$ on $\mathcal{S}$ such that the probability $q_1 := \mathbb{P}_{S \sim \mathcal{S}}[i \in S]$ is independent of $i \in [h]$, and the probability $q_2 := \mathbb{P}_{S \in \mathcal{S}}[i, j \in S]$ is independent of $i, j \in [h]$. A special case is when $\mathcal{S}$ is the set of all subsets of size k, and $D_{\mathcal{S}}$ is the uniform distribution on $\mathcal{S}$. For every $S \in \mathcal{S}$ there is a distribution say D_S on $(\mathbb{R}^{\geq 0})^h$ which is supported on vectors whose support is contained in S and which is uncorrelated for pairs of coordinates $i, j \in S$. Further, we assume that the distributions D_S are such that each coordinate x_i^* is compactly supported over an interval $[a(h), b(h)]$, where $a(h)$ and $b(h)$ are independent of both i and S but will be functions of h. Moreover, $m_1(h) := \mathbb{E}_{x^* \sim D_S}[x_i^*]$, and $m_2(h) := \mathbb{E}_{x^* \sim D_S}[x_i^{*2}]$ are assumed to be independent of both i and S but allowed to depend on h. For ease of notation henceforth we will keep

the h dependence of these variables implicit and refer to them as a, b, m_1 and m_2. All of our results will hold in the special case when a, b, m_1, m_2 are constants (no dependence on h).

4.3 Main Results

4.3.1 Recovery of the support of the sparse code by a layer of ReLUs

First we prove the following theorem which precisely quantifies the sense in which a layer of ReLU gates is able to recover the support of the sparse code when the weight matrix of the deep net is close to the original dictionary. We recall that the size of the support of the sparse vector x^* is $k = h^p$ for some $0 < p < 1$. We also recall the parameters a, b as defining the support of the marginal distribution of each coordinate of x^* and m_1 is the expected value of this marginal distribution (recall that none of these depend on the coordinate or the actual support). These parameters will be referenced in the results below.

Theorem 4.3.1.

We recall from equation 1.3 that our autoencoding neural net under consideration is mapping,

$$\begin{aligned} &\mathbb{R}^n \to \mathbb{R}^n \\ &\mathbf{y} \mapsto \mathrm{W}^T \mathbf{r} \text{ where } \mathbf{r} = \mathrm{ReLU}\left(\mathrm{W}\mathbf{y} - \epsilon\right) \end{aligned}$$

where the h columns of $W^\top$ are denoted as $\{W_i \in \mathbb{R}^n \mid i = 1, \ldots, h\}$

Let each column of $W^\top$ be within a δ-ball of the corresponding column of A^*, where $\delta = O\left(h^{-p-\nu^2}\right)$ for some $\nu > 0$, such that $p + \nu^2 < \xi$ (where $h^{-\xi}$ is the coherence parameter). We further assume that $a = \Omega\left(bh^{-\nu^2}\right)$. Let the bias of the hidden layer of the autoencoder as given above, be $\epsilon = 2m_1 k\left(\delta + \frac{\mu}{\sqrt{n}}\right)$. Then $r_i \neq 0$ if $i \in \mathrm{supp}(x^*)$, and $r_i = 0$ if $i \notin \mathrm{supp}(x^*)$ with probability at least $1 - \exp\left(-\frac{2h^p m_1^2}{(b-a)^2}\right)$ (with respect to the distribution on x^*).

As long as $\frac{h^p m_1^2}{(b-a)^2}$ is large, i.e., an increasing function of h, we can interpret this as saying that the probability of the adverse event is small, and we have successfully achieved support recovery at the hidden layer in the limit of large sparse code dimension.

4.3.2 Asymptotic Criticality of the Autoencoder around A^*

In this work we analyze the following standard squared loss function for the autoencoder,

$$L = \frac{1}{2}||\hat{y} - y||^2 \tag{4.1}$$

If we consider a generative model in which A^* is a square, orthogonal matrix and x^* is a non-negative vector (not necessarily sparse), it is easily seen that the standard squared reconstruction error loss function for the autoencorder has a global minimum at $W = A^{*\top}$. In our generative model, however, A^* is an incoherent and overcomplete dictionary.

Theorem 4.3.2. (**The Main Theorem**) Assume that the hypotheses of Theorem 4.3.1 hold, and $p < \min\{\frac{1}{2}, \nu^2\}$ (and hence $\xi > 2p$). Further, assume the distribution parameters satisfy $\exp\left(\frac{h^p m_1^2}{2(b-a)^2}\right)$ is superpolynomial in h (which holds, for example, when m_1, a, b are $O(1)$). Then for $i = 1, \ldots, h$,

$$\left\| \mathbb{E}\left[\frac{\partial L}{\partial W_i}\right] \right\|_2 \leq o\left(\frac{\max\{m_1^2, m_2\}}{h^{1-p}}\right).$$

Roadmap. We present the proof of the support recovery result, i.e., Theorem 4.3.1, in Section 4.4. Section 4.5 gives the proof of our main result, Theorem 4.3.2. The argument rests on Lemmas 4.5.1 and 4.5.2), whose proofs appear in Appendix 4.7 In Section 4.6, we run simulations to verify Theorem 4.3.2. We also run experiments that strongly suggest that the standard squared loss function has a local minimum in a neighborhood around A^*.

4.4 A Layer of ReLU Gates can Recover the Support of the Sparse Code (Proof of Theorem 4.3.1)

Most sparse coding algorithms are based on an alternating minimization approach, where one iteratively finds a sparse code based on the current estimate of the dictionary, and then uses the estimated sparse code to update the dictionary. The analogue of the sparse coding step in an autoencoder, is the passing through the hidden layer of activations of a certain affine transformation (W which behaves as the current estimate of the dictionary) of the input vectors. We show that under certain stochastic assumptions, the hidden layer of ReLU gates in an autoencoder recovers with high probability the support of the sparse vector which corresponds to the present input.

Proof of Theorem 4.3.1. From the model assumptions, we know that the dictionary A^* is incoherent,

and has unit norm columns. So, $|\langle A_i^*, A_j^* \rangle| \leq \frac{\mu}{\sqrt{n}}$ for all $i \neq j$, and $||A_i^*|| = 1$ for all i. This means that for $i \neq j$,

$$\begin{aligned} |\langle W_i, A_j^* \rangle| &= |\langle W_i - A_i^*, A_j^* \rangle| + |\langle A_i^*, A_j^* \rangle| \\ &\leq ||W_i - A_i^*||_2 ||A_j^*||_2 + \frac{\mu}{\sqrt{n}} \leq (\delta + \frac{\mu}{\sqrt{n}}) \end{aligned} \tag{4.2}$$

Otherwise for $i = j$,

$$\langle W_i, A_i^* \rangle = \langle W_i - A_i^*, A_i^* \rangle + \langle A_i^*, A_i^* \rangle = \langle W_i - A_i^*, A_i^* \rangle + 1,$$

and thus,

$$1 - \delta \leq \langle W_i, A_i^* \rangle \leq 1 + \delta, \tag{4.3}$$

where we use the fact that $|\langle W_i - A_i^*, A_i^* \rangle| \leq \delta$.

Let $y = A^* \mathbf{x}^*$ and let S be the support of $\mathbf{x}^*$. Then we define the input to the ReLU activation $Q - \epsilon = W\mathbf{y} - \epsilon$ as

$$Q_i = \sum_{j \in S} \langle W_i, A_j^* \rangle x_j^* = \langle W_i, A_i^* \rangle x_i^* \mathbf{1}_{i \in S} + \sum_{j \in S \setminus i} \langle W_i, A_j^* \rangle x_j^* = \langle W_i, A_i^* \rangle x_i^* \mathbf{1}_{i \in S} + Z_i.$$

First we try to get bounds on Q_i when $i \in \text{supp}(x^*)$. From our assumptions on the distribution of x_i^* we have, $0 \leq a \leq x_i^* \leq b$ and $\mathbb{E}[x_i^*] = m_1$ for all i in the support of x^*. For $i \in \text{supp}(x^*)$,

$$Q_i = \langle W_i, A_i^* \rangle x_i^* + Z_i \implies Q_i \geq (1 - \delta)a + Z_i$$

where we use (4.3). Using (4.2), Z_i has the following bounds:

$$-bk\left(\delta + \frac{\mu}{\sqrt{n}}\right) \leq Z_i \leq bk\left(\delta + \frac{\mu}{\sqrt{n}}\right)$$

Plugging in the lower bound for Z_i and the proposed value for the bias, we get

$$Q_i - \epsilon \geq (1 - \delta)a - bk\left(\delta + \frac{\mu}{\sqrt{n}}\right) - 2m_1 k\left(\delta + \frac{\mu}{\sqrt{n}}\right)$$

For $Q_i - \epsilon \geq 0$, we need:

$$a \geq \frac{(b+2m_1)\left(\delta + \frac{\mu}{\sqrt{n}}\right)k}{1-\delta}$$

Now plugging in the values for the various quantities, $\frac{\mu}{\sqrt{n}} = h^{-\xi}$ and $k = h^p$ and $\delta = O\left(h^{-p-\nu^2}\right)$, if we have $a = \Omega\left(bh^{-\nu^2}\right)$, then $Q_i - \epsilon \geq 0$.

Now, for $i \notin \text{supp}(x^*)$ we would like to analyze the following probability:

$$\Pr[Q_i - \epsilon \geq 0 | i \notin \text{supp}(x^*)]$$

We first simplify the quantity $\Pr[Q_i - \epsilon \geq 0 | i \notin \text{supp}(x^*)]$ as follows

$$\Pr[Q_i \geq \epsilon | i \notin \text{supp}(x^*)] = \Pr[Z_i \geq \epsilon] = \Pr\left[\sum_{j \in S \setminus i} \langle W_i, A_j^* \rangle x_j^* \geq \epsilon\right]$$

We recall that we had assumed that for every possible support S (of x^*) the distribution D_S on $(\mathbb{R}^{\geq 0})^h$, which is supported on vectors whose support is contained in S, is s.t the random variables corresponding to coordinates $i, j \in S$ are uncorrelated. Now using the Chernoff's bound, we can obtain

$$\begin{aligned}
\Pr[Z_i \geq \epsilon] &\leq \inf_{t \geq 0} e^{-t\epsilon} \mathbb{E}\left[\prod_{j \in S \setminus i} \left[e^{t\langle W_i, A_j^* \rangle x_j^*}\right]\right] = \inf_{t \geq 0} e^{-t\epsilon} \prod_{j \in S \setminus i} \mathbb{E}\left[e^{t\langle W_i, A_j^* \rangle x_j^*}\right] \\
&\leq \inf_{t \geq 0} e^{-t\epsilon} \mathbb{E}^k \left[e^{t\left(\delta + \frac{\mu}{\sqrt{n}}\right)x_j^*}\right] \\
&\leq \inf_{t \geq 0} e^{-t\epsilon} \left(e^{t\left(\delta + \frac{\mu}{\sqrt{n}}\right)m_1} e^{\frac{t^2\left(\delta + \frac{\mu}{\sqrt{n}}\right)^2 (b-a)^2}{8}}\right)^k
\end{aligned}$$

where the second inequality follows from (4.2) and the fact that t and x_i^* are both nonnegative, and the third inequality follows from Hoeffding's Lemma. Next, we also have

$$\begin{aligned}
\Pr[Z_i \geq \epsilon] &\leq \inf_{t \geq 0} e^{-t\left(\epsilon - k\left(\delta + \frac{\mu}{\sqrt{n}}\right)m_1\right) + t^2 \frac{k}{8}\left(\delta + \frac{\mu}{\sqrt{n}}\right)^2 (b-a)^2} \\
&= e^{-\frac{(\epsilon - k(\delta + \frac{\mu}{\sqrt{n}})m_1)^2}{\frac{k}{2}(\delta + \frac{\mu}{\sqrt{n}})^2 (b-a)^2}}.
\end{aligned}$$

Finally, since $k = h^p$ and $\epsilon = 2m_1 k\left(\delta + \frac{\mu}{\sqrt{n}}\right)$, we have

$$\exp\left(-\frac{2(\epsilon - km_1(\delta + \frac{\mu}{\sqrt{n}}))^2}{h^p(\delta + \frac{\mu}{\sqrt{n}})^2(b-a)^2}\right) = \exp\left(-\frac{2h^p m_1^2}{(b-a)^2}\right)$$

□

4.5 Criticality of a neighborhood of A^* (Proof of Theorem 4.3.2)

It turns out that the expectation of the full gradient of the loss function (4.1) is difficult to analyze directly. Hence corresponding to the true gradient with respect to the $i^{\text{th}}-$column of $W^\top$ we create a proxy, denoted by $\widehat{\nabla_i L}$, by replacing in the expression for the true expectation $\nabla_i L = \mathbb{E}\left[\frac{\partial L}{\partial W_i}\right]$ every occurrence of the random variable $\mathbf{1}_{W_i^\top y - \epsilon_i \geq 0} = \text{Th}(W_i^\top y - \epsilon_i) = \text{Th}(W_i^\top A^* x^* - \epsilon_i)$ by the indicator random variable $\mathbf{1}_{i \in \text{supp}(x^*)}$. This proxy is shown to be a good approximant of the expected gradient in the following lemma.

Lemma 4.5.1. Assume that the hypotheses of Theorem 4.3.1 hold and additionally let b be bounded by a polynomial in h. Then we have for each i (indexing the columns of $W^\top$),

$$\left|\left|\widehat{\nabla_i L} - \mathbb{E}\left[\frac{\partial L}{\partial W_i}\right]\right|\right|_2 \leq \text{poly}(h)\exp\left(-\frac{h^p m_1^2}{2(b-a)^2}\right)$$

Proof. This lemma has been proven in Section 4.A of the Appendix. □

Lemma 4.5.2.

Assume that the hypotheses of Theorem 4.3.1 hold, and $p < \min\{\frac{1}{2}, \nu^2\}$ (and hence $\xi > 2p$). Then for each i indexing the columns of $W^\top$, there exist real valued functions α_i and β_i, and a vector e_i such that $\widehat{\nabla_i L} = \alpha_i W_i - \beta_i A_i^* + e_i$, and

$$\alpha_i = \Theta(m_2 h^{p-1}) + o(m_1^2 h^{p-1})$$
$$\beta_i = \Theta(m_2 h^{p-1}) + o(m_1^2 h^{p-1})$$
$$\alpha_i - \beta_i = o(\max\{m_1^2, m_2\} h^{p-1})$$
$$||e_i||_2 = o(\max\{m_1^2, m_2\} h^{p-1})$$

Proof. In subsection 4.5.1 we first get explicit forms of the above defined quantities α_i, β_i and e_i. Then the proof is completed by estimating them which is done in Appendix 4.B □

With the above asymptotic results, we are in a position to assemble the proof of Theorem 4.3.2.

Proof of Theorem 4.3.2. Consider any i indexing the columns of $W^\top$. Recall the definition of the proxy gradient $\widehat{\nabla_i L}$ at the beginning of this section. Let us define $\gamma_i = \widehat{\nabla_i L} - \mathbb{E}\left[\frac{\partial L}{\partial W_i}\right]$. Using α_i, β_i and e_i as defined in Lemma 4.5.2, we can write the expectation of the true gradient as, $\mathbb{E}\left[\frac{\partial L}{\partial W_i}\right] = \alpha_i W_i - \beta_i A_i^* + e_i - \gamma_i$. Further, by Lemma 4.5.1,

$$\|\gamma_i\| \leq \text{poly}(h)\exp\left(-\frac{h^p m_1^2}{2(b-a)^2}\right).$$

Since $\exp\left(\frac{h^p m_1^2}{2(b-a)^2}\right)$ is superpolynomial in h, we obtain

$$\begin{aligned}
\left\|\mathbb{E}\left[\frac{\partial L}{\partial W_i}\right]\right\|_2 &= ||\alpha_i W_i - \beta_i A_i^* + e_i - \gamma_i||_2 \\
&= ||\alpha_i(W_i - A_i^*) + (\alpha_i - \beta_i)A_i^* + e_i - \gamma_i||_2 \\
&\leq |\alpha_i|\|W_i - A_i^*\|_2 + |\alpha_i - \beta_i| + ||e_i - \gamma_i||_2 \\
&\leq \frac{\Theta(m_2 h^{p-1})}{h^{2p+\theta^2}} + o(\max\{m_1^2, m_2\}h^{p-1}) \\
&+ o(\max\{m_1^2, m_2\}h^{p-1}) \\
&= o(\max\{m_1^2, m_2\}h^{p-1})
\end{aligned}$$

□

4.5.1 Simplifying the proxy gradient of the autoencoder under the sparse-coding generative model - to get explicit forms of the coefficients α, β and e as required towards proving Lemma 4.5.2

To recap we imagine being given as input signals $y \in \mathbb{R}^n$ (imagined as column vectors), which are generated from an overcomplete dictionary $A^* \in \mathbb{R}^{n \times h}$ of fixed incoherence. Let $x^* \in \mathbb{R}^h$ (imagined as column vectors) be the sparse code that generates y. The model of the autoencoder that we now have is $\hat{y} = W^\top \text{ReLU}(Wy - \epsilon)$. W is a $h \times n$ matrix and the i^{th} column of $W^\top$ is to be denoted as the column vector W_i.

Using the above notation the squared loss of the autoencoder is $\frac{1}{2}||\hat{y} - y||^2$. But we introduce a dummy constant $D = 1$ to be multiplied to y because this helps read the complicated equations that

would now follow. This marker helps easily spot those terms which depend on the sensing of x^* (those with a factor of D) as opposed to the terms which are "purely" dependent on the neural net (those without the factor of D). Thus we think of the squared loss L of our autoencoder as,

$$L = \frac{1}{2}||\hat{y} - Dy||^2 = \frac{1}{2}(W^\top \text{ReLU}(Wy - \epsilon) - Dy)^\top (W^\top \text{ReLU}(Wy - \epsilon) - Dy) = \frac{1}{2} f^T f$$

where we have defined $f \in \mathbb{R}^n$ as,

$$f = W^\top \text{ReLU}(Wy - \epsilon) - Dy$$

Then we have,

$$J_{W_i}(f)_{ab} = \frac{\partial f_a}{\partial W_{ib}} = \text{ReLU}(W_i^\top y - \epsilon)\delta_{ab} + \text{Th}(W_i^T y - \epsilon) W_{ia} y_b$$

In the form of a $n \times n$ derivative matrix this means,

$$J_{W_i}(f) = \left[\frac{\partial f_a}{\partial W_{ib}}\right] = \text{ReLU}(W_i^\top y - \epsilon) I + \text{Th}(W_i^\top y - \epsilon) W_i y^\top$$

This helps us write,

$$\begin{aligned}
\frac{\partial L}{\partial W_i} &= J_{W_i}(f))^\top f \\
&= (\text{ReLU}(W_i^\top y - \epsilon) I + \text{Th}(W_i^\top y - \epsilon) W_i y^\top)^\top [W^\top \text{ReLU}(Wy - \epsilon) - Dy] \\
&= \text{Th}(W_i^\top y - \epsilon_i) \left[(W_i^\top y - \epsilon_i) I + y W_i^\top\right] \left(\sum_{j=1}^{h} \text{ReLU}(W_j^\top y - \epsilon_j) W_j - Dy\right)
\end{aligned}$$

Now going over to the proxy gradient $\widehat{\nabla_i L}$ corresponding to this term and we define the vector G_i as,

$$\begin{aligned}
\widehat{\nabla_i L} &= \mathbb{E}_{S \in \mathcal{S}} \left[\mathbf{1}_{i \in S} \times \mathbb{E}_{x_S^*} \left[\left[(W_i^\top y - \epsilon_i) I + y W_i^\top\right] \left(\sum_{j \in S} (W_j^\top y - \epsilon_j) W_j - Dy\right)\right]\right] \\
&= \mathbb{E}_{S \in \mathcal{S}} \left[\mathbf{1}_{i \in S} \times G_i\right]
\end{aligned}$$

Thus we have,

$$
\begin{aligned}
G_i &= \mathbb{E}_{x_S^*}\left[\left[(W_i^\top A^* x^* - \epsilon_i)I + (A^* x^*)W_i^\top\right]\left(\sum_{j\in S}(W_j^\top A^* x^* - \epsilon_j)W_j - DA^* x^*\right)\right] \\
&= \underbrace{\mathbb{E}_{x_S^*}\left[(W_i^\top A^* x^* - \epsilon_i)\left(\sum_{j\in S}(W_j^\top A^* x^* - \epsilon_j)W_j - DA^* x^*\right)\right]}_{\text{Term 1}} \\
&+ \underbrace{\mathbb{E}_{x_S^*}\left[(A^* x^*)W_i^\top\left(\sum_{j\in S}(W_j^\top A^* x^* - \epsilon_j)W_j - DA^* x^*\right)\right]}_{\text{Term 2}}
\end{aligned}
$$

which can be decomposed into the following convenient parts,

$$
\begin{aligned}
G_i = &\underbrace{\mathbb{E}_{x_S^*}\left[\sum_{j\in S}\epsilon_i\epsilon_j W_j - \sum_{j,k\in S}\epsilon_i(W_j^\top A_k^*)W_j x_k^* - \sum_{j,k\in S}\epsilon_j(W_i^\top A_k^*)W_j x_k^* + \sum_{j,k,l\in S}(W_i^\top A_k^*)(W_j^\top A_l^*)W_j x_l^* x_k^*\right]}_{\text{From Term 1}} \\
&+ \underbrace{\mathbb{E}_{x_S^*}\left[-D\sum_{j,k\in S}(W_i^\top A_k^*)A_j^* x_k^* x_j^* + D\sum_{j\in S}\epsilon_i A_j^* x_j^*\right]}_{\text{From Term 1}} + \underbrace{\mathbb{E}_{x_S^*}\left[-D\sum_{j,k\in S}(A_k^{*\top} W_i)A_j^* x_k^* x_j^*\right]}_{\text{From Term 2}} \\
&+ \underbrace{\mathbb{E}_{x_S^*}\left[-\sum_{j,k\in S}\epsilon_j A_k^*(W_i^\top W_j)x_k^*\right]}_{\text{From Term 2}} + \underbrace{\mathbb{E}_{x_S^*}\left[\sum_{j,k,l\in S}(W_i^\top W_j)(W_j^\top A_l^*)A_k^* x_k^* x_l^*\right]}_{\text{From Term 2}}
\end{aligned}
$$

Now we invoke the distributional assumption about i.i.d sampling of the coordinates for a fixed support and the definition of m_1 and m_2 to write, $\mathbb{E}_{x_S^*}[x_i^* x_j^*] = \mathbb{E}^2_{x_S^*}[x_i^*] = m_1^2$ for all $i \neq j$ and for $i = j$, $m_2 = \mathbb{E}_{x_S^*}[x_i^* x_j^*]$. Thus we get,

$$
\begin{aligned}
G_i = & \underbrace{\sum_{j\in S} \epsilon_i\epsilon_j W_j - m_1 \sum_{j,k\in S} (W_j^\top A_k^*) W_j \epsilon_i - m_1 \sum_{j,k\in S} \epsilon_j (W_i^\top A_k^*) W_j}_{G_i^1 \text{ From Term 1}} \\
& \underbrace{+ m_2 \sum_{j,k\in S} (W_i^\top A_k^*)(W_j^\top A_k^*) W_j + m_1^2 \sum_{\substack{j,k,l\in S \\ k\neq l}} (W_i^\top A_k^*)(W_j^\top A_l^*) W_j}_{G_i^2 \text{ From Term 1}} \\
& \underbrace{+ \left[-Dm_1^2 \sum_{\substack{j,k\in S \\ j\neq k}} (W_i^\top A_k^*) A_j^* - Dm_2 \sum_{j\in S} (W_i^\top A_j^*) A_j^* + m_1 D \sum_{j\in S} \epsilon_i A_j^* \right]}_{G_i^3 \text{ From Term 1}} \\
& \underbrace{- \left[Dm_1^2 \sum_{\substack{j,k\in S \\ j\neq k}} (A_k^{*\top} W_i) A_j^* + Dm_2 \sum_{j\in S} (A_j^{*\top} W_i) A_j^* \right]}_{G_i^4 \text{ From Term 2}} \\
& \underbrace{- m_1 \left[\sum_{j,k\in S} \epsilon_j (W_i^\top W_j) A_k^* \right] + \left[m_2 \sum_{j,k\in S} (W_i^\top W_j)(W_j^\top A_k^*) A_k^* + m_1^2 \sum_{\substack{j,k,l\in S \\ k\neq l}} (W_i^\top W_j)(W_j^\top A_l^*) A_k^* \right]}_{G_i^5 \text{ From Term 2}}
\end{aligned}
$$

Each term in the above sum is a vector. Now we separate out from the sums the terms which are in the directions of W_i or A_i^* and the rest. We remember that this is being under the condition that $i \in S$. To make this easy to read we do this separation for each line of the above equation separately in a different equation block. Also inside every block we do the separation for each summation term in a separate line.

$$
\begin{aligned}
G_i^1 &= \sum_{j \in S} \epsilon_i \epsilon_j W_j - m_1 \sum_{j,k \in S} (W_j^\top A_k^*) W_j \epsilon_i - m_1 \sum_{j,k \in S} \epsilon_j (W_i^\top A_k^*) W_j \\
&= \left[\epsilon_i^2 W_i + \sum_{\substack{j \in S \\ j \neq i}} \epsilon_i \epsilon_j W_j \right] \\
&\quad - m_1 \left[\sum_{k \in S} \epsilon_i (W_i^\top A_k^*) W_i + \sum_{\substack{j,k \in S \\ j \neq i}} (W_j^\top A_k^*) W_j \epsilon_i \right] \\
&\quad - m_1 \left[\sum_{k \in S} \epsilon_i (W_i^\top A_k^*) W_i + \sum_{\substack{j,k \in S \\ j \neq i}} \epsilon_j (W_i^\top A_k^*) W_j \right]
\end{aligned}
$$

$$
\begin{aligned}
G_i^2 &= m_2 \sum_{j,k \in S} (W_i^\top A_k^*)(W_j^\top A_k^*) W_j + m_1^2 \sum_{\substack{j,k,l \in S \\ k \neq l}} (W_i^\top A_k^*)(W_j^\top A_l^*) W_j \\
&= m_2 \left[\sum_{k \in S} (W_i^\top A_k^*)(W_i^\top A_k^*) W_i + \sum_{\substack{j,k \in S \\ j \neq i}} (W_i^\top A_k^*)(W_j^\top A_k^*) W_j \right] \\
&\quad + m_1^2 \left[\sum_{\substack{k,l \in S \\ k \neq l}} (W_i^\top A_k^*)(W_i^\top A_l^*) W_i + \sum_{\substack{j,k,l \in S \\ j \neq i \\ k \neq l}} (W_i^\top A_k^*)(W_j^\top A_l^*) W_j \right]
\end{aligned}
$$

$$
\begin{aligned}
G_i^3 &= -D\left[m_1^2 \sum_{\substack{j,k\in S\\ j\neq k}} (W_i^\top A_k^*)A_j^* + m_2 \sum_{j\in S}(W_i^\top A_j^*)A_j^* - m_1 \sum_{j\in S}\epsilon_i A_j^*\right] \\
&= -D\left[m_1^2 \sum_{\substack{k\in S\\ k\neq i}}(W_i^\top A_k^*)A_i^* + m_1^2 \sum_{\substack{j,k\in S\\ j\neq i\\ j\neq k}} (W_i^\top A_k^*)A_j^*\right] \\
&\quad - D\left[m_2(W_i^\top A_i^*)A_i^* + m_2 \sum_{\substack{j\in S\\ j\neq i}}(W_i^\top A_j^*)A_j^*\right] \\
&\quad - D\left[-m_1\epsilon_i A_i^* - m_1 \sum_{\substack{j\in S\\ j\neq i}}\epsilon_i A_j^*\right]
\end{aligned}
$$

$$
\begin{aligned}
G_i^4 &= -\left[Dm_1^2 \sum_{\substack{j,k\in S\\ j\neq k}} (A_k^{*\top} W_i)A_j^* + Dm_2 \sum_{j\in S}(A_j^{*\top} W_i)A_j^*\right] \\
&= -D\left[m_1^2 \sum_{\substack{k\in S\\ k\neq i}}(A_k^{*\top} W_i)A_i^* + m_1^2 \sum_{\substack{j,k\in S\\ j\neq k\\ j\neq i}} (A_k^{*\top} W_i)A_j^*\right] \\
&\quad - D\left[m_2(A_i^{*\top} W_i)A_i^* + m_2 \sum_{\substack{j\in S\\ j\neq i}}(A_j^{*\top} W_i)A_j^*\right]
\end{aligned}
$$

$$G_i^5 = -m_1 \left[\sum_{j,k \in S} \epsilon_j (W_i^\top W_j) A_k^* \right] + \left[m_2 \sum_{j,k \in S} (W_i^\top W_j)(W_j^\top A_k^*) A_k^* + m_1^2 \sum_{\substack{j,k,l \in S \\ k \neq l}} (W_i^\top W_j)(W_j^\top A_l^*) A_k^* \right]$$

$$= -m_1 \sum_{j \in S} \epsilon_j (W_i^\top W_j) A_i^* - m_1 \sum_{\substack{j,k \in S \\ k \neq i}} \epsilon_j (W_i^\top W_j) A_k^*$$

$$+ m_2 \sum_{j \in S} (W_i^\top W_j)(W_j^\top A_i^*) A_i^* + m_2 \sum_{\substack{j,k \in S \\ k \neq i}} (W_i^\top W_j)(W_j^\top A_k^*) A_k^*$$

$$+ m_1^2 \sum_{\substack{j,l \in S \\ l \neq i}} (W_i^\top W_j)(W_j^\top A_l^*) A_i^* + m_1^2 \sum_{\substack{j,k,l \in S \\ k \neq i,l}} (W_i^\top W_j)(W_j^\top A_l^*) A_k^*$$

Thus combining the $G_i^1, \ldots, G_i^5$ above we have, $\widehat{\nabla_i L} = \alpha_i W_i - \beta_i A_i^* + e_i$ where,

$$\alpha_i = \mathbb{E}_{S \in \mathcal{S}} \left[\mathbf{1}_{i \in S} \times \left\{ m_2 \sum_{k \in S} (W_i^\top A_k^*)(W_i^\top A_k^*) + m_1^2 \sum_{\substack{k,l \in S \\ k \neq l}} (W_i^\top A_k^*)(W_i^\top A_l^*) - 2m_1 \sum_{k \in S} \epsilon_i (W_i^\top A_k^*) + \epsilon_i^2 \right\} \right]$$

$$\beta_i = \mathbb{E}_{S \in \mathcal{S}} \left[\mathbf{1}_{i \in S} \times \left\{ 2Dm_1^2 \sum_{\substack{k \in S \\ k \neq i}} (W_i^\top A_k^*) + 2Dm_2 (W_i^\top A_i^*) - Dm_1 \epsilon_i + m_1 \sum_{j \in S} \epsilon_j (W_i^\top W_j) \right.\right.$$

$$\left.\left. - m_2 \sum_{j \in S} (W_i^\top W_j)(W_j^\top A_i^*) - m_1^2 \sum_{\substack{j,l \in S \\ l \neq i}} (W_i^\top W_j)(W_j^\top A_l^*) \right\} \right]$$

$$e_i = \mathbb{E}_{S \in \mathcal{S}} \left[\mathbf{1}_{i \in S} \times \left\{ \sum_{\substack{j \in S \\ j \neq i}} \epsilon_i \epsilon_j W_j - m_1 \sum_{\substack{j,k \in S \\ j \neq i}} \epsilon_i (W_j^\top A_k^*) W_j - m_1 \sum_{\substack{j,k \in S \\ j \neq i}} \epsilon_j (W_i^\top A_k^*) W_j \right.\right.$$

$$+ m_2 \sum_{\substack{j,k \in S \\ j \neq i}} (W_i^\top A_k^*)(W_j^\top A_k^*) W_j + m_1^2 \sum_{\substack{j,k,l \in S \\ j \neq i \\ k \neq l}} (W_i^\top A_k^*)(W_j^\top A_l^*) W_j$$

$$- 2Dm_1^2 \sum_{\substack{j,k \in S \\ j \neq i \\ j \neq k}} (W_i^\top A_k^*) A_j^* - 2Dm_2 \sum_{\substack{j \in S \\ j \neq i}} (W_i^\top A_j^*) A_j^* + Dm_1 \sum_{\substack{j \in S \\ j \neq i}} \epsilon_i A_j^*$$

$$\left.\left. - m_1 \sum_{\substack{j,k \in S \\ k \neq i}} \epsilon_j (W_i^\top W_j) A_k^* + m_2 \sum_{\substack{j,k \in S \\ k \neq i}} (W_i^\top W_j)(W_j^\top A_k^*) A_k^* + m_1^2 \sum_{\substack{j,k,l \in S \\ k \neq i,l}} (W_i^\top W_j)(W_j^\top A_l^*) A_k^* \right\} \right]$$

Thus we have laid the groundwork of finding a convenient decomposition of the proxy-gradient in terms of the quantities α_i, β_i and e_i. Now we can go over to Appendix 4.B where their magnitudes are estimated towards completing the proof of Lemma 4.5.2.

4.6 Simulations

We conduct some experiments on synthetic data in order to check whether the gradient norm is indeed small within the columnwise δ-ball of A^*. We also make some observations about the landscape of the squared loss function, which has implications for being able to recover the ground-truth dictionary A^*.

Data Generation Model We generate random gaussian dictionaries (A^*) of size $n \times h$ where $n = 50$, and $h = 256, 512, 1024, 2048$ and 4096. For each h, we generate a dataset containing $N = 5000$ sparse vectors with h^p non-zero entries, for various $p \in [0.01, 0.5]$. In our experiments, the coherence parameter ξ was approximately 0.1. The support of each sparse vector x^* is drawn uniformly from all sets of indices of size h^p, and the non-zero entries in the sparse vectors are drawn from a uniform distribution between $a = 1$ and $b = 10$. Once we have generated the sparse vectors, we collect them in a matrix $X^* \in \mathbb{R}^{h \times N}$ and then compute the signals $Y = A^*X^*$. We set up the autoencoder as defined through equation 1.3. We analyze the squared loss function in (4.1) and its gradient with respect to a column of W through their empirical averages over the signals in Y.

Results Once we have generated the data, we compute the empirical average of the gradient of the loss function in (4.1) at 200 random points which are columnwise $\frac{\delta}{2} = \frac{1}{2h^{2p}}$ away from A^*. We average the gradient over the 200 points which are all at the same distance from A^*, and compare the average column norm of the gradient to h^{p-1}. Our experimental results shown in Table 4.6.1 demonstrate that the average column norm of the gradient is of the order of h^{p-1} (and thus falling with h for any fixed p) as expected from Theorem 4.3.2.

h \ p	0.01	0.02	0.05	0.1	0.2
256	(0.0137, 0.0041)	(0.0138, 0.0044)	(0.0126, 0.0052)	(0.0095, 0.0068)	(0.0284, 0.0118)
512	(0.0058, 0.0021)	(0.0058, 0.0022)	(0.0054, 0.0027)	(0.0071, 0.0036)	(0.0104, 0.0068)
1024	(0.0025, 0.0010)	(0.0024, 0.0011)	(0.0026, 0.0014)	(0.0079, 0.0020)	(0.0078, 0.0039)
2048	(0.0011, 0.0005)	(0.0012, 0.0006)	(0.0025, 0.0007)	(0.0031, 0.0010)	(0.0032, 0.0022)
4096	(0.0006, 0.0003)	(0.0012, 0.0003)	(0.0013, 0.0004)	(0.0026, 0.0006)	(0.0020, 0.0013)

h \ p	0.3	0.5
256	(0.0464, 0.0206)	(0.0343, 0.0625)
512	(0.0214, 0.0127)	(0.0028, 0.0442)
1024	(0.0099, 0.0078)	(0.00, 0.0313)
2048	(0.0036, 0.0048)	(0.00, 0.0221)
4096	(0.0008, 0.0030)	(0.00, 0.0156)

TABLE 4.6.1: Average gradient norm for points that are columnwise $\frac{\delta}{2}$ away from A^*. For each h and p we report $\left(||\mathbb{E}\left[\frac{\partial L}{\partial W_i}\right]||, h^{p-1}\right)$. We note that the gradient norm and h^{p-1} are of the same order, and for any fixed p the gradient norm is decreasing with h as expected from Theorem 4.3.2

We also plot the squared loss of the autoencoder along a randomly chosen direction to understand the geometry of the landscape of the loss function around A^*. We draw a matrix ΔW from a standard normal distribution, and normalize its columns. We then plot $f(t) = L((A^* + t\Delta W)^\top)$, as well as the gradient norm averaged over all the columns. For purposes of illustration, we show these plots for $p = 0.01, 0.1, 0.3$. The plots for $h = 256$ are in Figure 4.6.1, and those for $h = 4096$ in Figure 4.6.2. From the plots for $p = 0.01$ and 0.1, we can observe that the loss function value, and the gradient norm keep decreasing as we get close to A^*. Figure 4.6.1 and 4.6.2 are representative of the shapes obtained for every direction, ΔW that we checked. This suggests that A^* might conveniently lie at the bottom of a well in the landscape of the loss function. For the value of $p = 0.3$, (which is much larger than the coherence parameter ξ), Theorem 4.3.1 is no longer valid. We see that the value of the loss function decreases a little as we move away from A^*, and then increases. We suspect that A^* is here in a region where $\mathrm{ReLU}(A^{*\top} y - \epsilon) = 0$, which means the function is flat in a small neighborhood of A^*.

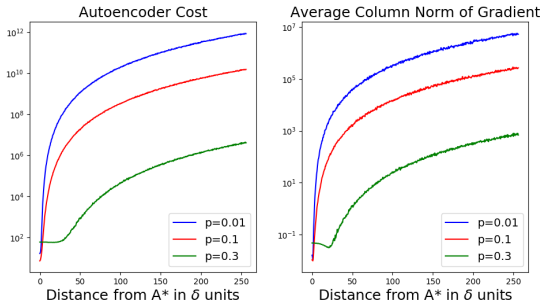


FIGURE 4.6.1: Loss function plot for $h = 256, n = 50$

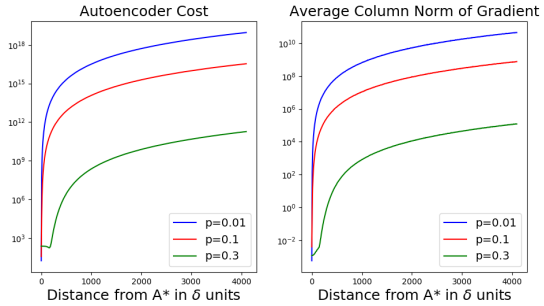


FIGURE 4.6.2: Loss function plot for $h = 4096, n = 50$

We also tried to minimize the squared loss of the autoencoder using gradient descent. In these experiments, we initialized $W^\top$ far away from A^* (precisely at a columnwise distance of $\frac{h}{5} \times \delta$), and did gradient descent until the gradient norm dropped below a factor of 2×10^{-5} of the initial norm of the gradient. We then computed the average columnwise distance between $W_{\text{final}}^\top$ and A^*, and report the % decrease in the average columnwise distance from the initial point. These results are reported in Table 4.6.2 below. These experiments suggest that there is a neighborhood of A^* (the radius of which

is increasing with h), such that gradient descent initialized at the edge of that neighborhood, greatly reduces the average columnwise distance between $W^\top$ and A^*.

h	$p = 0.05$	$p = 0.1$
256	97.7%	96.9%
512	98.6%	98.2%
1024	99%	98.8%
2048	99.2%	99%
4096	99.4%	99.2%

TABLE 4.6.2: Fraction of initial columnwise distance covered by the gradient descent procedure

4.7 Conclusion

In this chapter we have undertaken a rigorous analysis of the loss function of the squared loss of an autoencoder when the data is assumed to be generated by sensing of sparse high dimensional vectors by an overcomplete dictionary. **We have shown that the expected gradient of this loss function is very close to zero in a neighborhood of the generating overcomplete dictionary.**

Our simulations complement this theoretical result by providing further empirical support. Firstly, they show that the gradient norm in this $\delta-$ball of A^* indeed falls with h and is of the same order as $\frac{1}{h^{1-p}}$ as expected from our proof. Secondly, the experiments also strongly suggest ranges of values of h and p where A^* is a local minima of this loss function and that it has a neighborhood where the reconstruction error is low.

This suggests sparse coding problems can be solved by training autoencoders using gradient descent based algorithms. Further, recent investigations have led to the conjecture/belief that many important unsupervised learning tasks, e.g. recognizing handwritten digits, are sparse coding problems in disguise (Makhzani and Frey, 2013; Makhzani and Frey, 2015). Thus, our results could shed some light on the observed phenomenon that gradient descent based algorithms train autoencoders to low reconstruction error for natural data sets, like MNIST.

It remains to rigorously show whether a gradient descent algorithm can be initialized randomly (may be far away from A^*) and still be shown to converge to this neighborhood of critical points around the dictionary. Towards that it might be helpful to understand the structure of the Hessian outside this neighborhood. Since our analysis applies to the expected gradient, it remains to analyze the sample complexities where these nice results will become prominent.

The possibility also remains open that this standard loss or some other loss functions exist for the autoencoder with the provable property of having a global minima/minimum at the ground truth dictionary. We have mentioned one example of such in a special case (when A^* is square orthogonal and x^* is nonnegative) and even in this special case it remains open to find a provable optimization algorithm.

On the simulation front we have a couple of open challenges yet to be tackled. Firstly, it is left to find efficient implementations of the iterative update rule based on the exact gradient of the proposed loss function which has been given in (4.1). This would open up avenues for testing the power of this loss function on real data rather than the synthetic data used here. Secondly, a simulation of the main Theorem 4.3.2 that can probe deeper into its claim would need to be able to sample A^* for different h at a fixed value of the incoherence parameter ξ. This sampling question of A^* with these constraints is an unresolved one that is left for future work.

Autoencoders with more than one hidden layer have been used for unsupervised feature learning (Le, 2013) and recently there has been an analysis of the sparse coding performance of convolutional neural networks with one layer (Gilbert et al., 2017) and two layers of nonlinearities (Vardan, Romano, and Elad, 2016). The connections between neural networks and sparse coding has also been recently explored in Bora et al., 2017. It remains an exciting open avenue of research to try to do a similar study as in this work to determine if and how deeper architectures under the same generative model might provide better means of doing sparse coding.

Appendix To Chapter 4

4.A The proxy gradient is a good approximation of the true expectation of the gradient (Proof of Lemma 5.1)

Proof. To make it easy to present this argument let us abstractly think of the function f (defined for any $i \in \{1, 2, 3, .., h\}$) as $f(y, W, X) = \frac{\partial L}{\partial W_i}$ where we have defined the random variable $X = \text{Th}[W_i^T y - \epsilon_i]$. It is to be noted that because of the ReLU term and its derivative this function f has a dependency on $y = A^* x^*$ even outside its dependency through X. Let us define another random variable $Y = \mathbf{1}_{i \in \text{Support}(x^*)}$. Then we have,

$$
\begin{aligned}
&\left\| \mathbb{E}_{x^*}[f(y, W, X)] - \mathbb{E}_{x^*}[f(y, W, Y)] \right\|_{\ell_2} \\
\leq &\mathbb{E}_{x^*}[|f(y, W, X) - f(y, W, Y)|_{\ell_2}] \\
\leq &\mathbb{E}_{x^*}[|f(y, W, X)(\mathbf{1}_{X=Y} + \mathbf{1}_{X \neq Y}) - f(y, W, Y)(\mathbf{1}_{X=Y} + \mathbf{1}_{X \neq Y})|_{\ell_2}] \\
\leq &\mathbb{E}_{x^*}[|(f(y, W, X) - f(y, W, Y))|_{\ell_2} \mathbf{1}_{X \neq Y}] \\
\leq &\sqrt{\mathbb{E}_{x^*}[\left|f(y, W, X) - f(y, W, Y)\right|_2^2]} \sqrt{\mathbb{E}_{x^*}[\mathbf{1}_{X \neq Y}]}
\end{aligned}
$$

In the last step above we have used the Cauchy-Schwarz inequality for random variables. We recognize that $\mathbb{E}_{x^*}[f(y, W, Y)]$ is precisely what we defined as the proxy gradient $\widehat{\nabla_i L}$. Further for such W as in this lemma the support recovery theorem (Theorem 3.1) holds and that is precisely the statement that the term, $\mathbb{E}_{x^*}[\mathbf{1}_{X \neq Y}]$ is small. So we can rewrite the above inequality as,

$$
\left\| \mathbb{E}_{x^*}[\frac{\partial L}{\partial W_i}] - \widehat{\nabla_i L} \right\|_2 \leq \sqrt{\mathbb{E}_{x^*}[\left|f(y, W, X) - f(y, W, Y)\right|_2^2]} \exp\left(-\frac{h^p m_1^2}{2(b-a)^2} \right)
$$

We remember that f is a polynomial in h because its h dependency is through Frobenius norms of submatrices of W and ℓ_2 norms of projections of Wy. But the ℓ_∞ norm of the training vectors y (that is b) have been assumed to be bounded by $\text{poly}(h)$. Also we have the assumption that the columns of

$W^\top$ are within a $\frac{1}{h^{p+\nu^2}}-$ball of the corresponding columns of A^* which in turn is a $n \times h$ dimensional matrix of bounded norm because all its columns are normalized. So summarizing we have,

$$\left\| \mathbb{E}_{x^*}[\frac{\partial L}{\partial W_i}] - \widehat{\nabla_i L} \right\|_2 \leq \text{poly}(h) \exp\left(-\frac{h^p m_1^2}{2(b-a)^2} \right)$$

The above inequality immediately implies the claimed lemma. □

4.B The asymptotics of the coefficients of the gradient of the squared loss (Proof of Lemma 5.2)

We will pick up from where subsection 4.5.1 left and will now estimate bounds on each of the terms $\alpha_i, \beta_i, ||e_i||$, which were defined at the end of that segment. We will separate them as $\alpha_i = \tilde{\alpha_i} + \hat{\alpha_i}$ (similarly for the other terms). Where the tilde terms are those that come as a coefficient of m_2, and the hat terms are the ones that come as coefficient of m_1 or ϵ or both. (Note : Given the previous definitions of q_1 and q_2 it is obvious from context as to how the quantities q_i, q_{ij}, q_{ijk} and q_S mean and we shall use this notation in this Appendix.)

4.B.1 Estimating the m_2 dependent parts of the derivative

Since $||A_i^*|| = 1$ and W_i is being assumed to be within a $0 < \delta < 1$ ball of A_i^* we can use the following inequalities:

$$\begin{aligned}
||W_i|| &= ||W_i - A_i^* + A_i^*|| \leq ||W_i - A_i^*|| + ||A_i^*|| = \delta + 1 \\
||W_i|| &\geq 1 - \delta \\
\langle W_i, A_i^* \rangle &= \langle W_i - A_i^*, A_i^* \rangle + \langle A_i^*, A_i^* \rangle \leq ||W_i - A_i^*||||A_i^*|| + 1 \leq \delta + 1 \\
\langle W_i, A_i^* \rangle &\geq 1 - \delta \\
|\langle W_j, A_i^* \rangle| &= |\langle W_j - A_j^*, A_i^* \rangle + \langle A_j^*, A_i^* \rangle| \leq \frac{\mu}{\sqrt{n}} + ||W_j - A_j^*||||A_i^*|| = \frac{\mu}{\sqrt{n}} + \delta \\
|\langle W_i, W_j \rangle| &= |\langle W_i - A_i^*, W_j \rangle + \langle A_i^*, W_j \rangle| \leq \delta(1+\delta) + (\delta + \frac{\mu}{\sqrt{n}}) = \delta^2 + 2\delta + \frac{\mu}{\sqrt{n}} \\
\langle W_i, W_i \rangle &= ||W_i||^2 \geq (1-\delta)^2 \\
\langle W_i, W_i \rangle &= ||W_i||^2 \leq (1+\delta)^2
\end{aligned}$$

Bounding $\tilde{\beta}_i$

$$\tilde{\beta}_i = \mathbb{E}_{S\in\mathbb{S}}\left[\mathbf{1}_{i\in S}\left\{2Dm_2(W_i^\top A_i^*) - m_2\sum_{j\in S}(W_i^\top W_j)(W_j^\top A_i^*)\right\}\right]$$
$$= \mathbb{E}_{S\in\mathbb{S}}\left[\mathbf{1}_{i\in S}\left\{2Dm_2\langle W_i, A_i^*\rangle - m_2||W_i||^2\langle W_i, A_i^*\rangle - m_2\sum_{\substack{j\in S\\ j\neq i}}\langle W_i, W_j\rangle\langle W_j, A_i^*\rangle\right\}\right]$$

Evaluating the outer expectation we get,

$$\tilde{\beta}_i = \sum_{\{S\in\mathbb{S}:i\in S\}} q_S 2Dm_2\langle W_i, A_i^*\rangle - \sum_{\{S\in\mathbb{S}:i\in S\}} q_S m_2||W_i||^2\langle W_i, A_i^*\rangle - m_2\sum_{\substack{j=1\\ j\neq i}}^{h}\langle W_i, W_j\rangle\langle W_j, A_i^*\rangle \sum_{\{S\in\mathbb{S}:i,j\in S, i\neq j\}} q_S$$
$$= 2Dq_i m_2\langle W_i, A_i^*\rangle - q_i m_2||W_i||^2\langle W_i, A_i^*\rangle - m_2\sum_{\substack{j=1\\ j\neq i}}^{h} q_{ij}\langle W_i, W_j\rangle\langle W_j, A_i^*\rangle$$

Upper bounding the above we get,

$$\tilde{\beta}_i \leq 2Dm_2h^{p-1}(1+\delta) - m_2h^{p-1}(1-\delta)^3 + m_2h^{2p-1}\left(\delta + \frac{\mu}{\sqrt{n}}\right)\left(\delta^2 + 2\delta + \frac{\mu}{\sqrt{n}}\right)$$
$$= 2Dm_2h^{p-1}(1+h^{-p-\nu^2}) - m_2h^{p-1}(1-3h^{-p-\nu^2}+3h^{-2p-2\nu^2}-h^{-3p-3\nu^2})$$
$$+ m_2h^{2p-1}(h^{-3p-3\nu^2}+2h^{-2p-2\nu^2}+h^{-2p-2\nu^2-\xi}+3h^{-p-\nu^2-\xi}+h^{-2\xi}) \tag{4.4}$$

Similarly for the lower bound on β_i we get,

$$\tilde{\beta}_i \geq 2Dm_2h^{p-1}(1-\delta) - m_2h^{p-1}(1+\delta)^3 - m_2h^{2p-1}\left(\delta + \frac{\mu}{\sqrt{n}}\right)\left(\delta^2 + 2\delta + \frac{\mu}{\sqrt{n}}\right)$$
$$= 2Dm_2h^{p-1}(1-h^{-p-\nu^2}) - m_2h^{p-1}(1+3h^{-p-\nu^2}+3h^{-2p-2\nu^2}+h^{-3p-3\nu^2})$$
$$- m_2h^{2p-1}(h^{-3p-3\nu^2}+2h^{-2p-2\nu^2}+h^{-2p-2\nu^2-\xi}+3h^{-p-\nu^2-\xi}+h^{-2\xi}) \tag{4.5}$$

Thus for $0 < p < 2\xi$ and $D = 1$, we have $\beta = \Theta\left(m_2h^{p-1}\right)$

Bounding $\tilde{\alpha_i}$

$$\begin{aligned}
\tilde{\alpha_i} &= \mathbb{E}_{S\in\mathbb{S}}\left[\mathbf{1}_{i\in S}\left\{m_2\sum_{k\in S}(W_i^\top A_k^*)^2\right\}\right] \\
&= \mathbb{E}_{S\in\mathbb{S}}\left[\mathbf{1}_{i\in S}\left\{m_2\langle W_i, A_i^*\rangle^2 + m_2\sum_{\substack{k\in S\\k\neq i}}\langle W_i, A_k^*\rangle^2\right\}\right] \\
&= \sum_{\{S\in\mathbb{S}:i\in S\}} m_2\langle W_i, A_i^*\rangle^2 q_S + \sum_{\substack{k=1\\k\neq i}}^{h}\sum_{\{S\in\mathbb{S}:i,k\in S\}}\langle W_i, A_k^*\rangle^2 q_S \\
&= m_2\langle W_i, A_i^*\rangle^2\sum_{\{S\in\mathbb{S}:i\in S\}} q_S + m_2\sum_{\substack{k=1\\k\neq i}}^{h}\langle W_i, A_k^*\rangle^2\left(\sum_{\{S\in\mathbb{S}:i,k\in S,i\neq k\}} q_S\right) \\
&= q_i m_2\langle W_i, A_i^*\rangle^2 + m_2\sum_{\substack{k=1\\k\neq i}}^{h} q_{ik}\langle W_i, A_k^*\rangle^2 \\
&= h^{p-1}m_2\langle W_i, A_i^*\rangle^2 + m_2 h^{2p-1}\max\langle W_i, A_k^*\rangle^2
\end{aligned}$$

The above implies the following bounds,

$$h^{p-1}m_2(1-h^{-p-\nu^2})^2 \leq \tilde{\alpha_i} \leq h^{p-1}m_2(1+h^{-p-\nu^2})^2 + m_2h^{2p-1}(h^{-p-\nu^2}+h^{-\xi})^2 \tag{4.6}$$

As long as $0 < p < 2\xi$, $\tilde{\alpha_i} = \Theta\left(m_2h^{p-1}\right)$

Bounding $||\tilde{e_i}||_2$

$$\begin{aligned}
\tilde{e_i} = \mathbb{E}_{S\in\mathbb{S}}&\left[\mathbf{1}_{i\in S}\times\left\{m_2\sum_{\substack{j,k\in S\\j\neq i}}(W_i^\top A_k^*)(W_j^\top A_k^*)W_j + (-2D)m_2\sum_{\substack{j\in S\\j\neq i}}(W_i^\top A_j^*)A_j^*\right\}\right] \\
+\mathbb{E}_{S\in\mathbb{S}}&\left[\mathbf{1}_{i\in S}\times\left\{m_2\sum_{\substack{j,k\in S\\k\neq i}}(W_i^\top W_j)(W_j^\top A_k^*)A_k^*\right\}\right]
\end{aligned}$$

Expanding further over the summation of the j and the k indices we have,

$$\tilde{e_i} = \mathbb{E}_{S\in\mathbb{S}}\Bigg[\mathbf{1}_{i\in S}\times m_2\Bigg\{\sum_{j(=k)\in S\setminus i}(W_i^\top A_j^*)(W_j^\top A_j^*)W_j + \sum_{\substack{j\in S\setminus i\\ k\in S\setminus i,j}}(W_i^\top A_k^*)(W_j^\top A_k^*)W_j$$
$$+\sum_{\substack{j\in S\setminus i\\ k=i}}(W_i^\top A_i^*)(W_j^\top A_i^*)W_j\Bigg\}\Bigg]$$
$$+\mathbb{E}_{S\in\mathbb{S}}\left[\mathbf{1}_{i\in S}\times(-2D)m_2\left\{\sum_{\substack{j\in S\\ j\neq i}}(W_i^\top A_j^*)A_j^*\right\}\right]$$
$$+\mathbb{E}_{S\in\mathbb{S}}\Bigg[\mathbf{1}_{i\in S}\times m_2\Bigg\{\sum_{k(=j)\in S\setminus i}(W_i^\top W_k)(W_k^\top A_k^*)A_k^* + \sum_{\substack{k\in S\setminus i\\ j\in S\setminus i,k}}(W_i^\top W_j)(W_j^\top A_k^*)A_k^*$$
$$+\sum_{\substack{k\in S\setminus i\\ j=i}}(W_i^\top W_i)(W_i^\top A_k^*)A_k^*\Bigg\}\Bigg]$$

Expanding the above in terms of q_S we have,

$$\tilde{e_i} = m_2\Bigg\{\sum_{j=1,j\neq i}^{h}(W_i^\top A_j^*)(W_j^\top A_j^*)W_j\sum_{\{S\in\mathbb{S}:i,j\in S,i\neq j\}}q_S + \sum_{\substack{j,k=1\\ j\neq k\neq i}}^{h}(W_i^\top A_k^*)(W_j^\top A_k^*)W_j\sum_{\{S\in\mathbb{S}:i,j,k\in S,i\neq j\neq k\}}q_S$$
$$+\sum_{\substack{j=1\\ j\neq i}}^{h}(W_i^\top A_i^*)(W_j^\top A_i^*)W_j\sum_{\{S\in\mathbb{S}:i,j\in S,i\neq j\}}q_S\Bigg\}$$
$$+(-2D)m_2\left\{\sum_{\substack{j=1\\ j\neq i}}^{h}(W_i^\top A_j^*)A_j^*\sum_{\{S\in\mathbb{S}:i,j\in S,i\neq j\}}q_S\right\}$$
$$+m_2\Bigg\{\sum_{\substack{k=1\\ k\neq i}}^{h}(W_i^\top W_k)(W_k^\top A_k^*)A_k^*\sum_{\{S\in\mathbb{S}:i,k\in S,i\neq k\}}q_S + \sum_{\substack{j,k=1\\ j\neq i\neq k}}^{h}(W_i^\top W_j)(W_j^\top A_k^*)A_k^*\sum_{\{S\in\mathbb{S}:i,j,k\in S,i\neq j\neq k\}}q_S$$
$$+\sum_{\substack{k=1\\ k\neq i}}^{h}(W_i^\top W_i)(W_i^\top A_k^*)A_k^*\sum_{\{S\in\mathbb{S}:i,k\in S,i\neq k\}}q_S\Bigg\}$$

Expanding the q_S dependency in terms of q_{ij} and q_{ijk} we have,

$$\tilde{e_i} = m_2 \Bigg\{ \sum_{j=1, j\neq i}^{h} q_{ij}(W_i^\top A_j^*)(W_j^\top A_j^*)W_j + \sum_{\substack{j,k=1 \\ j\neq k\neq i}}^{h} q_{ijk}(W_i^\top A_k^*)(W_j^\top A_k^*)W_j$$
$$+ \sum_{\substack{j=1 \\ j\neq i}}^{h} q_{ij}(W_i^\top A_i^*)(W_j^\top A_i^*)W_j \Bigg\} + (-2D)m_2 \left\{ \sum_{\substack{j=1 \\ j\neq i}}^{h} q_{ij}(W_i^\top A_j^*)A_j^* \right\}$$
$$+ m_2 \Bigg\{ \sum_{\substack{k=1 \\ k\neq i}}^{h} q_{ik}(W_i^\top W_k)(W_k^\top A_k^*)A_k^* + \sum_{\substack{j,k=1 \\ j\neq i\neq k}}^{h} q_{ijk}(W_i^\top W_j)(W_j^\top A_k^*)A_k^*$$
$$+ \sum_{\substack{k=1 \\ k\neq i}}^{h} q_{ik}(W_i^\top W_i)(W_i^\top A_k^*)A_k^* \Bigg\}$$

Upper bounding the norm of this vector $\tilde{e}_i$ we get,

$$\begin{aligned}
||\tilde{e_i}|| &\leq m_2 h^{2p-1}\left(\delta + \frac{\mu}{\sqrt{n}}\right)(1+\delta)^2 + m_2 h^{3p-1}\left(\delta + \frac{\mu}{\sqrt{n}}\right)^2(1+\delta) \\
&+ m_2 h^{2p-1}\left(\delta + \frac{\mu}{\sqrt{n}}\right)(1+\delta)^2 + 2Dm_2 h^{2p-1}\left(\delta + \frac{\mu}{\sqrt{n}}\right) \\
&+ m_2 h^{2p-1}\left(\delta^2 + 2\delta + \frac{\mu}{\sqrt{n}}\right)(1+\delta) + m_2 h^{3p-1}\left(\delta^2 + 2\delta + \frac{\mu}{\sqrt{n}}\right)\left(\delta + \frac{\mu}{\sqrt{n}}\right) \\
&+ m_2 h^{2p-1}\left(\delta + \frac{\mu}{\sqrt{n}}\right)(1+\delta)^2 \\
&\leq m_2 h^{2p-1}(h^{-p-\nu^2} + 2h^{-2p-2\nu^2} + h^{-3p-3\nu^2} + 2h^{-p-\nu^2-\xi} + h^{-2p-2\nu^2-\xi} + h^{-\xi}) \\
&+ m_2 h^{3p-1}(h^{-2p-2\nu^2} + h^{-3p-3\nu^2} + 2h^{-p-\nu^2-\xi} + 2h^{-2p-2\nu^2-\xi} + h^{-2\xi} + h^{-p-\nu^2-2\xi}) \\
&+ m_2 h^{2p-1}(h^{-p-\nu^2} + 2h^{-2p-2\nu^2} + h^{-3p-3\nu^2} + 2h^{-p-\nu^2-\xi} + h^{-2p-2\nu^2-\xi} + h^{-\xi}) \\
&+ 2Dm_2 h^{2p-1}(h^{-p-\nu^2} + h^{-\xi}) \\
&+ m_2 h^{2p-1}(2h^{-p-\nu^2} + 3h^{-2p-2\nu^2} + h^{-3p-3\nu^2} + h^{-p-\nu^2-\xi} + h^{-\xi}) \\
&+ m_2 h^{3p-1}(2h^{-2p-2\nu^2} + h^{-3p-3\nu^2} + 3h^{-p-\nu^2-\xi} + h^{-2p-2\nu^2-\xi} + h^{-2\xi}) \\
&+ m_2 h^{2p-1}(h^{-p-\nu^2} + 2h^{-2p-2\nu^2} + h^{-3p-3\nu^2} + 2h^{-p-\nu^2-\xi} + h^{-2p-2\nu^2-\xi} + h^{-\xi}) \qquad (4.7)
\end{aligned}$$

If $D = 1$ and $0 < p < \xi$, we get $||\tilde{e_i}|| = o(m_2 h^{p-1})$

4.B.2 Estimating the m_1 dependent parts of the derivative

We continue working in the same regime for the W matrix as in the previous subsection. Hence the same inequalities as listed at the beginning of the previous subsection continue to hold and we use them to get the following bounds,

Bounding $\hat{\alpha}_i$

$$\begin{aligned}
\hat{\alpha}_i &= \mathbb{E}_{S\in\mathbb{S}}\left[\mathbf{1}_{i\in S}\times\left\{m_1^2\sum_{\substack{k,l\in S\\k\neq l}}(W_i^\top A_k^*)(W_i^\top A_l^*) - 2m_1\sum_{k\in S}\epsilon_i(W_i^\top A_k^*)+\epsilon_i^2\right\}\right]\\
&= \mathbb{E}_{S\in\mathbb{S}}\left[\mathbf{1}_{i\in S}\times\left\{m_1^2\sum_{\substack{k\in S\\k\neq i}}\langle W_i,A_k^*\rangle\langle W_i,A_i^*\rangle + m_1^2\sum_{\substack{l\in S\\l\neq i}}\langle W_i,A_i^*\rangle\langle W_i,A_l^*\rangle + m_1^2\sum_{\substack{k,l\in S\\k\neq l\\k\neq i\\l\neq i}}\langle W_i,A_k^*\rangle\langle W_i,A_l^*\rangle\right.\right.\\
&\quad\left.\left. - 2m_1\epsilon_i\langle W_i,A_i^*\rangle - 2m_1\sum_{\substack{k\in S\\k\neq i}}\epsilon_i\langle W_i,A_k^*\rangle+\epsilon_i^2\right\}\right]\\
&= 2m_1^2\sum_{\substack{k=1\\k\neq i}}^{h}\langle W_i,A_k^*\rangle\langle W_i,A_i^*\rangle\sum_{\{S\in\mathbb{S}:i,k\in S,k\neq i\}}q_S + m_1^2\sum_{\substack{k,l=1\\k\neq l\\k\neq i\\l\neq i}}^{h}\langle W_i,A_k^*\rangle\langle W_i,A_l^*\rangle\sum_{\{S\in\mathbb{S}:i,k,l\in S,k\neq i\neq l\}}q_S\\
&\quad - 2m_1\epsilon_i\langle W_i,A_i^*\rangle\sum_{\{S\in\mathbb{S}:i\in S\}}q_S - 2m_1\sum_{\substack{k=1\\k\neq i}}^{h}\epsilon_i\langle W_i,A_k^*\rangle\sum_{\{S\in\mathbb{S}:i,k\in S,k\neq i\}}q_S + \epsilon_i^2\sum_{\{S\in\mathbb{S}:i\in S\}}q_S\\
\implies \hat{\alpha}_i &= 2m_1^2\sum_{\substack{k=1\\k\neq i}}^{h}q_{ik}\langle W_i,A_k^*\rangle\langle W_i,A_i^*\rangle + m_1^2\sum_{\substack{k,l=1\\k\neq l\\k\neq i\\l\neq i}}^{h}q_{ikl}\langle W_i,A_k^*\rangle\langle W_i,A_l^*\rangle\\
&\quad - 2m_1q_i\epsilon_i\langle W_i,A_i^*\rangle - 2m_1\sum_{\substack{k=1\\k\neq i}}^{h}q_{ik}\epsilon_i\langle W_i,A_k^*\rangle + q_i\epsilon_i^2
\end{aligned}$$

We plugin $\epsilon_i = 2m_1h^p\left(\delta+\frac{\mu}{\sqrt{n}}\right)$ for $i=1,\ldots,h$

$$
\begin{aligned}
|\hat{\alpha_i}| &\leq 2m_1^2h^{2p-1}\left(\delta+\frac{\mu}{\sqrt{n}}\right)(1+\delta)+m_1^2h^{3p-1}\left(\delta+\frac{\mu}{\sqrt{n}}\right)^2+4m_1^2h^{2p-1}(1+\delta)\left(\delta+\frac{\mu}{\sqrt{n}}\right)\\
&+4m_1^2h^{3p-1}\left(\delta+\frac{\mu}{\sqrt{n}}\right)^2+4m_1^2h^{3p-1}\left(\delta+\frac{\mu}{\sqrt{n}}\right)^2\\
&=2m_1^2h^{2p-1}(h^{-p-\nu^2}+h^{-2p-2\nu^2}+h^{-p-\nu^2-\xi}+h^{-\xi})+m_1^2h^{3p-1}(h^{-2p-2\nu^2}+2h^{-p-\nu^2-\xi}+h^{-2\xi})\\
&+4m_1^2h^{2p-1}(h^{-p-\nu^2}+h^{-2p-2\nu^2}+h^{-\xi}+h^{-p-\nu^2-\xi})+4m_1^2h^{3p-1}(h^{-2p-2\nu^2}+2h^{-p-\nu^2-\xi}+h^{-2\xi})\\
&+4m_1^2h^{3p-1}(h^{-2p-2\nu^2}+2h^{-p-\nu^2-\xi}+h^{-2\xi})
\end{aligned}
$$

This means that if $p<\xi$, $|\hat{\alpha_i}|=o(m_1^2h^{p-1})$. Putting this together with the bounds obtained below equation 4.6, we get that $\alpha_i=\Theta(m_2h^{p-1})+o(m_1^2h^{p-1})$.

Bounding $\hat{\beta}_i$

$$
\begin{aligned}
\hat{\beta_i} &= \mathbb{E}_{S\in\mathbb{S}}\left[\mathbf{1}_{i\in S}\times\left\{2Dm_1^2\sum_{\substack{k\in S\\k\neq i}}(W_i^\top A_k^*)-Dm_1\epsilon_i+m_1\sum_{j\in S}\epsilon_j(W_i^\top W_j)-m_1^2\sum_{\substack{j,l\in S\\l\neq i}}(W_i^\top W_j)(W_j^\top A_l^*)\right\}\right]\\
&=2Dm_1^2\sum_{\substack{k=1\\k\neq i}}^{h}\langle W_i,A_k^*\rangle\sum_{\{S\in\mathbb{S}:i,k\in S,k\neq i\}}q_S-Dm_1\epsilon_i\sum_{\{S\in\mathbb{S}:i\in S\}}q_S+m_1\epsilon_i||W_i||^2\sum_{\{S\in\mathbb{S}:i\in S\}}q_S\\
&+m_1\sum_{j=1,j\neq i}^{h}\epsilon_j\langle W_i,W_j\rangle\sum_{\{S\in\mathbb{S}:i,j\in S,j\neq i\}}q_S-m_1^2\sum_{\substack{l=1\\l\neq i}}^{h}||W_i||^2\langle W_i,A_l^*\rangle\sum_{\{S\in\mathbb{S}:i,l\in S,l\neq i\}}q_S\\
&-m_1^2\sum_{\substack{l=1\\l\neq i}}^{h}\langle W_i,W_l\rangle\langle W_l,A_l^*\rangle\sum_{\{S\in\mathbb{S}:i,l\in S,l\neq i\}}q_S-m_1^2\sum_{\substack{j,l=1\\l\neq i\\j\neq l,i}}^{h}\langle W_i,W_j\rangle\langle W_j,A_l^*\rangle\sum_{\{S\in\mathbb{S}:i,j,l\in S,l\neq i\neq i\}}q_S\\
&=2Dm_1^2\sum_{\substack{k=1\\k\neq i}}^{h}q_{ik}\langle W_i,A_k^*\rangle-Dm_1\epsilon_iq_i+m_1\epsilon_i||W_i||^2q_i+m_1\sum_{j=1,j\neq i}^{h}\epsilon_jq_{ij}\langle W_i,W_j\rangle\\
&-m_1^2\sum_{\substack{l=1\\l\neq i}}^{h}||W_i||^2\langle W_i,A_l^*\rangle q_{il}-m_1^2\sum_{\substack{l=1\\l\neq i}}^{h}\langle W_i,W_l\rangle\langle W_l,A_l^*\rangle q_{il}-m_1^2\sum_{\substack{j,l=1\\l\neq i\\j\neq l,i}}^{h}\langle W_i,W_j\rangle\langle W_j,A_l^*\rangle q_{ijl}
\end{aligned}
$$

We plugin $\epsilon_i=2m_1h^p\left(\delta+\frac{\mu}{\sqrt{n}}\right)$ for $i=1,\ldots,h$

$$
\begin{aligned}
|\hat{\beta}_i| &\leq 4Dm_1^2h^{2p-1}\left(\delta+\frac{\mu}{\sqrt{n}}\right)+2m_1^2h^{2p-1}\left(\delta+\frac{\mu}{\sqrt{n}}\right)(1+\delta)^2+2m_1^2h^{3p-1}\left(\delta+\frac{\mu}{\sqrt{n}}\right)\left(\delta^2+2\delta+\frac{\mu}{\sqrt{n}}\right)\\
&+m_1^2h^{2p-1}(1+\delta)^2\left(\delta+\frac{\mu}{\sqrt{n}}\right)+m_1^2h^{2p-1}\left(\delta^2+2\delta+\frac{\mu}{\sqrt{n}}\right)(1+\delta)\\
&+m_1^2h^{3p-1}\left(\delta^2+2\delta+\frac{\mu}{\sqrt{n}}\right)\left(\delta+\frac{\mu}{\sqrt{n}}\right)\\
&=4Dm_1^2h^{2p-1}(h^{-p-\nu^2}+h^{-\xi})\\
&+2m_1^2h^{2p-1}(h^{-p-\nu^2}+2h^{-2p-2\nu^2}+h^{-3p-3\nu^2}+h^{-\xi}+2h^{-p-\nu^2-\xi}+h^{-2p-2\nu^2-\xi})\\
&+2m_1^2h^{3p-1}(2h^{-2p-2\nu^2}+h^{-3p-3\nu^2}+3h^{-p-\nu^2-\xi}+h^{-2p-2\nu^2-\xi}+h^{-2\xi})\\
&+m_1^2h^{2p-1}(h^{-p-\nu^2}+2h^{-2p-2\nu^2}+h^{-3p-3\nu^2}+h^{-\xi}+2h^{-p-\nu^2-\xi}+h^{-2p-2\nu^2-\xi})\\
&+m_1^2h^{2p-1}(3h^{-2p-2\nu^2}+h^{-3p-3\nu^2}+h^{-p-\nu^2-\xi}+2h^{-p-\nu^2}+h^{-\xi})\\
&+m_1^2h^{3p-1}(2h^{-2p-2\nu^2}+h^{-3p-3\nu^2}+3h^{-p-\nu^2-\xi}+h^{-2p-2\nu^2-\xi}+h^{-2\xi})
\end{aligned}
$$

This means that if $p<\xi$, $|\hat{\beta}_i|=o(m_1^2h^{p-1})$. Putting this together with the bounds obtained below 4.4, we get that $\beta_i=\Theta(m_2h^{p-1})+o(m_1^2h^{p-1})$.

Bounding $||\hat{e_i}||_2$

$$
\begin{aligned}
\hat{e_i} &= \underbrace{\mathbb{E}_{S\in\mathfrak{S}}\left[\mathbf{1}_{i\in S}\times\left\{\sum_{\substack{j\in S\\ j\neq i}}\epsilon_i\epsilon_jW_j-m_1\sum_{\substack{j,k\in S\\ j\neq i}}(W_j^\top A_k^*)W_j\epsilon_i-m_1\sum_{\substack{j,k\in S\\ j\neq i}}\epsilon_j(W_i^\top A_k^*)W_j\right\}\right]}_{\hat{e_{i1}}}\\
&+\underbrace{\mathbb{E}_{S\in\mathfrak{S}}\left[\mathbf{1}_{i\in S}\times\left\{m_1^2\sum_{\substack{j,k,l\in S\\ j\neq i\\ k\neq l}}(W_i^\top A_k^*)(W_j^\top A_l^*)W_j\right\}\right]}_{\hat{e_{i2}}}\\
&+\underbrace{\mathbb{E}_{S\in\mathfrak{S}}\left[\mathbf{1}_{i\in S}\times\left\{-2Dm_1^2\sum_{\substack{j,k\in S\\ j\neq i\\ k\neq i}}(W_i^\top A_k^*)A_j^*+Dm_1\sum_{\substack{j\in S\\ j\neq i}}\epsilon_iA_j^*\right\}\right]}_{\hat{e_{i3}}}\\
&+\underbrace{\mathbb{E}_{S\in\mathfrak{S}}\left[\mathbf{1}_{i\in S}\times\left\{-m_1\sum_{\substack{j,k\in S\\ k\neq i}}\epsilon_j(W_i^\top W_j)A_k^*+m_1^2\sum_{\substack{j,k,l\in S\\ k\neq i,l}}(W_i^\top W_j)(W_j^\top A_l^*)A_k^*\right\}\right]}_{\hat{e_{i4}}}
\end{aligned}
$$

We estimate the different summands separately.

$$\hat{e_{i1}} = \mathbb{E}_{S\in\mathbf{S}}\left[\mathbf{1}_{i\in S}\times\left\{\sum_{\substack{j\in S\\ j\neq i}}\epsilon_i\epsilon_j W_j\right\}\right]$$
$$+\mathbb{E}_{S\in\mathbf{S}}\left[\mathbf{1}_{i\in S}\times(-m_1)\left\{\sum_{j(=k)\in S\setminus i}(W_j^\top A_j^*)W_j\epsilon_i+\sum_{\substack{j\in S\setminus i\\ k\in S\setminus i,j}}(W_j^\top A_k^*)W_j\epsilon_i+\sum_{\substack{j\in S\setminus i\\ k=i}}(W_j^\top A_i^*)W_j\epsilon_i\right\}\right]$$
$$+\mathbb{E}_{S\in\mathbf{S}}\left[\mathbf{1}_{i\in S}\times(-m_1)\left\{\sum_{j(=k)\in S\setminus i}\epsilon_j(W_i^\top A_j^*)W_j+\sum_{\substack{j\in S\setminus i\\ k\in S\setminus i,j}}\epsilon_j(W_i^\top A_k^*)W_j+\sum_{\substack{j\in S\setminus i\\ k=i}}\epsilon_j(W_i^\top A_i^*)W_j\right\}\right]$$

We substitute, $\epsilon = 2m_1h^p(h^{-p-\nu^2}+h^{-\xi})$ and for any two vectors $\mathbf{x}$ and $\mathbf{y}$ and any two scalars a and b we use the inequality, $||a\mathbf{x}+b\mathbf{y}||_2 \leq |a|_{max}||\mathbf{x}||_{2,max}+|b|_{max}||\mathbf{y}||_{2,max}$to get,

$$
\begin{aligned}
||\hat{e_{i1}}||_2 &\leq 4m_1^2h^{2p}\left(\delta+\frac{\mu}{\sqrt{n}}\right)^2\sum_{j=1,j\neq i}^{h}q_{ij}||W_j|| \\
&+2m_1^2h^p\left(\delta+\frac{\mu}{\sqrt{n}}\right)\left(\sum_{j=1,j\neq i}^{h}q_{ij}\langle W_j,A_j^*\rangle W_j+\sum_{j,k=1,j\neq i,k\neq i,j}^{h}q_{ijk}\langle W_j,A_k^*\rangle W_j\right. \\
&\left.+\sum_{j=1,j\neq i}^{h}q_{ij}\langle W_j,A_i^*\rangle W_j\right) \\
&+2m_1^2h^p\left(\delta+\frac{\mu}{\sqrt{n}}\right)\left(\sum_{j=1,j\neq i}^{h}q_{ij}\langle W_i,A_j^*\rangle W_j+\sum_{j,k=1,j\neq i,k\neq i,j}^{h}q_{ijk}\langle W_i,A_k^*\rangle W_j\right. \\
&\left.+\sum_{j=1,j\neq i}^{h}q_{ij}\langle W_i,A_i^*\rangle W_j\right) \\
\implies ||\hat{e_{i1}}||_2 &\leq 4m_1^2h^{2p}h^{2p-1}(1+\delta)\left(\delta+\frac{\mu}{\sqrt{n}}\right)^2 \\
&+2m_1^2h^p\left(\delta+\frac{\mu}{\sqrt{n}}\right)\left(h^{2p-1}(1+\delta)^2+h^{3p-1}\left(\delta+\frac{\mu}{\sqrt{n}}\right)(1+\delta)+h^{2p-1}\left(\delta+\frac{\mu}{\sqrt{n}}\right)(1+\delta)\right) \\
&+2m_1^2h^p\left(\delta+\frac{\mu}{\sqrt{n}}\right)\left(h^{2p-1}\left(\delta+\frac{\mu}{\sqrt{n}}\right)(1+\delta)+h^{3p-1}\left(\delta+\frac{\mu}{\sqrt{n}}\right)(1+\delta)+h^{2p-1}(1+\delta)^2\right) \\
\implies ||\hat{e_{i1}}||_2 &\leq 4m_1^2h^{4p-1}(1+\delta)\left(\delta+\frac{\mu}{\sqrt{n}}\right)^2 \\
&+2m_1^2h^{3p-1}\left(\delta+\frac{\mu}{\sqrt{n}}\right)(1+\delta)^2+2m_1^2h^{4p-1}\left(\delta+\frac{\mu}{\sqrt{n}}\right)^2(1+\delta) \\
&+2m_1^2h^{3p-1}\left(\delta+\frac{\mu}{\sqrt{n}}\right)^2(1+\delta) \\
&+2m_1^2h^{3p-1}\left(\delta+\frac{\mu}{\sqrt{n}}\right)^2(1+\delta)+2m_1^2h^{4p-1}\left(\delta+\frac{\mu}{\sqrt{n}}\right)^2(1+\delta) \\
&+2m_1^2h^{3p-1}\left(\delta+\frac{\mu}{\sqrt{n}}\right)(1+\delta)^2 \\
\implies ||\hat{e_{i1}}||_2 &\leq 8m_1^2h^{4p-1}(1+\delta)\left(\delta+\frac{\mu}{\sqrt{n}}\right)^2+4m_1^2h^{3p-1}\left(\delta+\frac{\mu}{\sqrt{n}}\right)(1+\delta)^2 \\
&+4m_1^2h^{3p-1}\left(\delta+\frac{\mu}{\sqrt{n}}\right)^2(1+\delta)
\end{aligned}
$$

$$
\begin{aligned}
\implies ||\hat{e_{i1}}||_2 &\leq 8m_1^2h^{4p-1}(h^{-2p-2\nu^2} + h^{-3p-3\nu^2} + 2h^{-p-\nu^2-\xi} + 2h^{-2p-2\nu^2-\xi} + h^{-p-\nu^2-2\xi} + h^{-2\xi}) \\
&+ 4m_1^2h^{3p-1}(h^{-p-\nu^2} + h^{-3p-3\nu^2} + 2h^{-2p-2\nu^2} + h^{-\xi} + h^{-2p-2\nu^2-\xi} + 2h^{-p-\nu^2-\xi}) \\
&+ 4m_1^2h^{3p-1}(h^{-2p-2\nu^2} + h^{-3p-3\nu^2} + 2h^{-p-\nu^2-\xi} + 2h^{-2p-2\nu^2-\xi} + h^{-p-\nu^2-2\xi} + h^{-2\xi}) \\
&= 8m_1^2h^{p-1}(h^{p-2\nu^2} + h^{-3\nu^2} + 2h^{p-\nu^2+p-\xi} + 2h^{-2\nu^2+p-\xi} + h^{-\nu^2+2p-2\xi} + h^{3p-2\xi}) \\
&+ 4m_1^2h^{p-1}(h^{p-\nu^2} + h^{-p-3\nu^2} + 2h^{-2\nu^2} + h^{2p-\xi} + h^{-2\nu^2-\xi} + 2h^{-\nu^2+p-\xi}) \\
&+ 4m_1^2h^{p-1}(h^{-2\nu^2} + h^{-p-3\nu^2} + 2h^{-\nu^2+p-\xi} + 2h^{-2\nu^2-\xi} + h^{-\nu^2+p-2\xi} + h^{2p-2\xi})
\end{aligned}
$$

From the above it follows that, $||\hat{e_{i1}}||_2 = o(m_1^2h^{p-1})$ for $p < \nu^2$ and $2p < \xi$

And now we start to estimate $\hat{e_{i2}}$

$$
\begin{aligned}
\hat{e_{i2}} &= \mathbb{E}_{S\in\mathbb{S}}\left[\mathbf{1}_{i\in S}\times m_1^2\left\{\sum_{\substack{j,k,l\in S\\ j\neq i\\ k\neq l}}(W_i^\top A_k^*)(W_j^\top A_l^*)W_j\right\}\right] \\
&= \mathbb{E}_{S\in\mathbb{S}}\Bigg[\mathbf{1}_{i\in S}\times m_1^2\Bigg\{\sum_{\substack{j\in S\\ j\neq i}}(W_i^\top A_j^*)(W_j^\top A_i^*)W_j+\sum_{\substack{j,k\in S\\ k\neq j\neq i}}(W_i^\top A_k^*)(W_j^\top A_i^*)W_j \\
&\quad+\sum_{\substack{j\in S\\ j\neq i}}(W_i^\top A_i^*)(W_j^\top A_j^*)W_j \\
&\quad+\sum_{\substack{j,l\in S\\ l\neq j\neq i}}(W_i^\top A_i^*)(W_j^\top A_l^*)W_j+\sum_{\substack{j,l\in S\\ l\neq j\neq i}}(W_i^\top A_j^*)(W_j^\top A_l^*)W_j+\sum_{\substack{j,k\in S\\ k\neq j\neq i}}(W_i^\top A_k^*)(W_j^\top A_j^*)W_j \\
&\quad+\sum_{\substack{j,k,l\in S\\ l\neq k\neq j\neq i}}(W_i^\top A_k^*)(W_j^\top A_l^*)W_j\Bigg\}\Bigg] \\
\implies \hat{e_{i2}} &= m_1^2\Bigg\{\sum_{\substack{j=1\\ j\neq i}}^{h}q_{ij}(W_i^\top A_j^*)(W_j^\top A_i^*)W_j+\sum_{\substack{j,k=1\\ k\neq j\neq i}}^{h}q_{ijk}(W_i^\top A_k^*)(W_j^\top A_i^*)W_j \\
&\quad+\underbrace{\sum_{\substack{j=1\\ j\neq i}}^{h}q_{ij}(W_i^\top A_i^*)(W_j^\top A_j^*)W_j}_{\mathbf{a}} \\
&\quad+\sum_{\substack{j,l=1\\ l\neq j\neq i}}^{h}q_{ijl}(W_i^\top A_i^*)(W_j^\top A_l^*)W_j+\sum_{\substack{j,l=1\\ l\neq j\neq i}}^{h}q_{ijl}(W_i^\top A_j^*)(W_j^\top A_l^*)W_j+\sum_{\substack{j,k=1\\ k\neq j\neq i}}^{h}q_{ijk}(W_i^\top A_k^*)(W_j^\top A_j^*)W_j \\
&\quad+\sum_{\substack{j,k,l\in S\\ l\neq k\neq j\neq i}}q_{ijkl}(W_i^\top A_k^*)(W_j^\top A_l^*)W_j\Bigg\} \\
\implies ||\hat{e_{i2}}|| &\leq m_1^2\Bigg\{h^{2p-1}\left(\delta+\frac{\mu}{\sqrt{n}}\right)^2(1+\delta)+h^{3p-1}\left(\delta+\frac{\mu}{\sqrt{n}}\right)^2(1+\delta)+||\mathbf{a}|| \\
&\quad+h^{3p-1}\left(\delta+\frac{\mu}{\sqrt{n}}\right)(1+\delta)^2+h^{3p-1}\left(\delta+\frac{\mu}{\sqrt{n}}\right)^2(1+\delta)+h^{3p-1}\left(\delta+\frac{\mu}{\sqrt{n}}\right)(1+\delta)^2 \\
&\quad+h^{4p-1}\left(\delta+\frac{\mu}{\sqrt{n}}\right)^2(1+\delta)\Bigg\}
\end{aligned}
$$

$$
\begin{aligned}
\implies ||\hat{e_{i2}}|| \leq m_1^2 \Big\{ & h^{2p-1}(h^{-2p-2\nu^2} + h^{-3p-3\nu^2} + 2h^{-p-\nu^2-\xi} + 2h^{-2p-2\nu^2-\xi} + h^{-p-\nu^2-2\xi} + h^{-2\xi}) \\
& + h^{3p-1}(h^{-2p-2\nu^2} + h^{-3p-3\nu^2} + 2h^{-p-\nu^2-\xi} + 2h^{-2p-2\nu^2-\xi} + h^{-p-\nu^2-2\xi} + h^{-2\xi}) \\
& + ||\mathbf{a}|| \\
& + h^{3p-1}(h^{-p-\nu^2} + h^{-3p-3\nu^2} + 2h^{-2p-2\nu^2} + h^{-2p-2\nu^2-\xi} + 2h^{-p-\nu^2-\xi} + h^{-\xi}) \\
& + h^{3p-1}(h^{-2p-2\nu^2} + h^{-3p-3\nu^2} + 2h^{-p-\nu^2-\xi} + 2h^{-2p-2\nu^2-\xi} + h^{-p-\nu^2-2\xi} + h^{-2\xi}) \\
& + h^{3p-1}(h^{-p-\nu^2} + h^{-3p-3\nu^2} + 2h^{-2p-2\nu^2} + h^{-2p-2\nu^2-\xi} + 2h^{-p-\nu^2-\xi} + h^{-\xi}) \\
& + h^{4p-1}(h^{-2p-2\nu^2} + h^{-3p-3\nu^2} + 2h^{-p-\nu^2-\xi} + 2h^{-2p-2\nu^2-\xi} + h^{-p-\nu^2-2\xi} + h^{-2\xi}) \Big\} \\
\implies ||\hat{e_{i2}}|| \leq m_1^2 \Big\{ & h^{p-1}(h^{-p-2\nu^2} + h^{-2p-3\nu^2} + 2h^{-\nu^2-\xi} + 2h^{-p-2\nu^2-\xi} + h^{-\nu^2-2\xi} + h^{p-2\xi}) \\
& + h^{p-1}(h^{-2\nu^2} + h^{-p-3\nu^2} + 2h^{-\nu^2+p-\xi} + 2h^{-2\nu^2-\xi} + h^{-\nu^2+p-2\xi} + h^{2p-2\xi}) \\
& + ||\mathbf{a}|| \\
& + h^{p-1}(h^{p-\nu^2} + h^{-p-3\nu^2} + 2h^{-2\nu^2} + h^{-2\nu^2-\xi} + 2h^{-\nu^2+p-\xi} + h^{2p-\xi}) \\
& + h^{p-1}(h^{-2\nu^2} + h^{-p-3\nu^2} + 2h^{-\nu^2+p-\xi} + 2h^{-2\nu^2-\xi} + h^{-\nu^2+p-2\xi} + h^{2p-2\xi}) \\
& + h^{p-1}(h^{p-\nu^2} + h^{-2p-3\nu^2} + 2h^{-2\nu^2} + h^{-2\nu^2-\xi} + 2h^{-\nu^2+p-\xi} + h^{2p-\xi}) \\
& + h^{p-1}(h^{p-2\nu^2} + h^{-3\nu^2} + 2h^{p-\nu^2+p-\xi} + 2h^{-2\nu^2+p-\xi} + h^{-\nu^2+2p-2\xi} + h^{3p-2\xi}) \Big\}
\end{aligned}
$$

Now let us find a bound for $||\mathbf{a}||$.

$$
\begin{aligned}
\mathbf{a} &= \sum_{\substack{j=1 \\ j \neq i}}^{h} q_{ij}(W_i^\top A_i^*)(W_j^\top A_j^*)W_j \\
&= \langle W_i, A_i^* \rangle q_{ij} W_{-j}^\top \text{diag}(W_{-j} A_{-j}^*)
\end{aligned}
$$

Where A_{-j}^* is the dictionary A^* with the jth column set to zero, W_{-j} is the dictionary W with the jth row set to zero, and $\text{diag}(W_{-j}A_{-j}^*)$ is the h-dimensional vector containing the diagonal elements of the matrix $W_{-j}A_{-j}^*$. We also make use of the distributional assumption that q_{ij} is the same for all i, j

in order to pull q_{ij} out of the sum.

$$
\begin{aligned}
||\mathbf{a}||_2 &= h^{2p-2}\langle W_i, A_i^*\rangle ||W_{-j}^\top \text{diag}(W_{-j}A_{-j}^*)||_2 \\
&\leq h^{2p-2}(1+\delta)||W_{-j}^\top||_2||\text{diag}(W_{-j}A_{-j}^*)||_2 \\
&\leq h^{2p-2}(1+\delta)^2 h^{1/2}\sqrt{\lambda_{\max}(W_{-j}^\top W_{-j})} \\
&\leq h^{2p-2}(1+\delta)^2 h^{1/2}\sqrt{h\left(\delta^2+2\delta+\frac{\mu}{\sqrt{n}}\right)+(1+\delta)^2} \\
&= h^{p-1}\sqrt{h^{2p-2}\times h\times(1+\delta)^4\times\left(h\left(\delta^2+2\delta+\frac{\mu}{\sqrt{n}}\right)+(1+\delta)^2\right)} \\
&= h^{p-1}\sqrt{h^{2p-1}\times(1+h^{-p-\nu^2})^4\times\left(h(h^{-2p-2\nu^2}+2h^{-p-\nu^2}+h^{-\xi})+(1+h^{-p-\nu^2})^2\right)} \\
&= h^{p-1}\sqrt{(1+h^{-p-\nu^2})^4\times(h^{-2\nu^2}+2h^{p-\nu^2}+h^{2p-\xi}+h^{2p-1}(1+h^{-p-\nu^2})^2)}
\end{aligned}
$$

Here $||W_{-j}^\top||_2$ is the spectral norm of $W_{-j}^\top$, and is the top singular value of the matrix. We use Gershgorin's Circle theorem to bound the top eigenvalue of $W_{-j}^\top W_{-j}$ by its maximum row sum.

If $p < \frac{\xi}{2}$, $p < \frac{1}{2}$, and $p < \nu^2$, then $||\hat{e_{i2}}|| = o(m_1^2 h^{p-1})$

And now we start to estimate $\hat{e_{i3}}$ as follows.

$$
\begin{aligned}
\hat{e_{i3}} &= \mathbb{E}_{S\in\mathbb{S}}\left[\mathbf{1}_{i\in S}\times\left\{Dm_1\sum_{\substack{j\in S\\ j\neq i}}\epsilon_i A_j^* - 2Dm_1^2\sum_{\substack{j,k\in S\\ j\neq i\\ k\neq i}}(W_i^\top A_k^*)A_j^*\right\}\right] \\
&= \mathbb{E}_{S\in\mathbb{S}}\left[\mathbf{1}_{i\in S}\times\left\{Dm_1\sum_{\substack{j\in S\\ j\neq i}}\epsilon_i A_j^* - 2Dm_1^2\sum_{\substack{j\in S\\ j\neq i}}(W_i^\top A_j^*)A_j^* - 2Dm_1^2\sum_{\substack{j,k\in S\\ k\neq j\neq i}}(W_i^\top A_k^*)A_j^*\right\}\right] \\
&= Dm_1\sum_{\substack{j=1\\ j\neq i}}^{h}\epsilon_i A_j^*\sum_{\{S\in\mathbb{S}:i,j\in S,i\neq j\}}q_S - 2Dm_1^2\sum_{\substack{j=1\\ j\neq i}}^{h}(W_i^\top A_j^*)A_j^*\sum_{\{S\in\mathbb{S}:i,j\in S,i\neq j\}}q_S \\
&- 2Dm_1^2\sum_{\substack{j,k=1\\ k\neq j\neq i}}^{h}(W_i^\top A_k^*)A_j^*\sum_{\{S\in\mathbb{S}:i,j,k\in S,i\neq j\neq k\}}q_S \\
&= Dm_1\sum_{\substack{j=1\\ j\neq i}}^{h}q_{ij}\epsilon_i A_j^* - 2Dm_1^2\sum_{\substack{j=1\\ j\neq i}}^{h}q_{ij}(W_i^\top A_j^*)A_j^* - 2Dm_1^2\sum_{\substack{j,k=1\\ k\neq j\neq i}}^{h}q_{ijk}(W_i^\top A_k^*)A_j^*
\end{aligned}
$$

We plugin $\epsilon_i = 2m_1 h^p\left(\delta+\frac{\mu}{\sqrt{n}}\right)$ for $i = 1,\ldots,h$

$$
\begin{aligned}
||\hat{e_{i3}}|| &\leq 2Dm_1^2h^{3p-1}\left(\delta+\frac{\mu}{\sqrt{n}}\right)+2Dm_1^2h^{2p-1}\left(\delta+\frac{\mu}{\sqrt{n}}\right)+2Dm_1^2h^{3p-1}\left(\delta+\frac{\mu}{\sqrt{n}}\right) \\
&= 4Dm_1^2h^{3p-1}(h^{-p-\nu^2}+h^{-\xi})+2Dm_1^2h^{2p-1}(h^{-p-\nu^2}+h^{-\xi}) \\
&= 4Dm_1^2h^{p-1}(h^{p-\nu^2}+h^{2p-\xi})+2Dm_1^2h^{p-1}(h^{-\nu^2}+h^{p-\xi})
\end{aligned}
$$

This means for $D=1$, $p<\nu^2$ and $p<\frac{\xi}{2}$, we have $||\hat{e_{i3}}|| = o(m_1^2h^{p-1})$

And now we start to estimate $\hat{e_{i4}}$ as follows.

$$
\begin{aligned}
\hat{e_{i4}} &= \mathbb{E}_{S\in\mathbb{S}}\left[\mathbf{1}_{i\in S}\times\left\{-m_1\sum_{\substack{j,k\in S\\ k\neq i}}\epsilon_j(W_i^\top W_j)A_k^* + m_1^2\sum_{\substack{j,k,l\in S\\ k\neq i,l}}(W_i^\top W_j)(W_j^\top A_l^*)A_k^*\right\}\right]\\
&= \mathbb{E}_{S\in\mathbb{S}}\left[\mathbf{1}_{i\in S}\times(-m_1)\left\{\sum_{k(=j)\in S\setminus i}\epsilon_k(W_i^\top W_k)A_k^* + \sum_{\substack{j\in S\setminus i\\ k\in S\setminus i,j}}\epsilon_j(W_i^\top W_j)A_k^* + \sum_{\substack{k\in S\setminus i\\ j=i}}\epsilon_j(W_i^\top W_i)A_k^*\right\}\right]\\
&+ \mathbb{E}_{S\in\mathbb{S}}\left[\mathbf{1}_{i\in S}\times m_1^2\left\{\sum_{\substack{j,k,l\in S\\ k\neq i,l}}(W_i^\top W_j)(W_j^\top A_l^*)A_k^*\right\}\right]\\
&= \mathbb{E}_{S\in\mathbb{S}}\left[\mathbf{1}_{i\in S}\times(-m_1)\left\{\sum_{k(=j)\in S\setminus i}\epsilon_k(W_i^\top W_k)A_k^* + \sum_{\substack{j\in S\setminus i\\ k\in S\setminus i,j}}\epsilon_j(W_i^\top W_j)A_k^* + \sum_{\substack{k\in S\setminus i\\ j=i}}\epsilon_j(W_i^\top W_i)A_k^*\right\}\right]\\
&+ \mathbb{E}_{S\in\mathbb{S}}\Bigg[\mathbf{1}_{i\in S}\times m_1^2\Bigg\{\sum_{\substack{k\in S\\ k\neq i}}(W_i^\top W_i)(W_i^\top A_i^*)A_k^* + \sum_{\substack{k\in S\\ k\neq i}}(W_i^\top W_k)(W_k^\top A_i^*)A_k^* + \sum_{\substack{j,k\in S\\ j\neq k\neq i}}(W_i^\top W_j)(W_j^\top A_i^*)A_k^*\\
&+ \sum_{\substack{k,l\in S\\ \neq k\neq i}}(W_i^\top W_i)(W_i^\top A_l^*)A_k^* + \sum_{\substack{k,l\in S\\ l\neq k\neq i}}(W_i^\top W_k)(W_k^\top A_l^*)A_k^* + \sum_{\substack{k,l\in S\\ l\neq k\neq i}}(W_i^\top W_l)(W_l^\top A_l^*)A_k^*\\
&+ \sum_{\substack{j,k,l\in S\\ j\neq k\neq l\neq i}}(W_i^\top W_j)(W_j^\top A_l^*)A_k^*\Bigg\}\Bigg]
\end{aligned}
$$

$$
\begin{aligned}
\hat{e_{i4}} &= (-m_1)\left\{\sum_{k=1,k\neq i}^{h} q_{ik}\epsilon_k(W_i^\top W_k)A_k^* + \sum_{\substack{j,k=1\\ j\neq k\neq i}}^{h} q_{ijk}\epsilon_j(W_i^\top W_j)A_k^* + \sum_{\substack{k=1\\ k\neq i}}^{h} q_{ik}\epsilon_i(W_i^\top W_i)A_k^*\right\}\\
&+ m_1^2\Bigg\{\underbrace{\sum_{\substack{k=1\\ k\neq i}}^{h} q_{ik}(W_i^\top W_i)(W_i^\top A_i^*)A_k^*}_{\mathbf{b}} + \sum_{\substack{k=1\\ k\neq i}}^{h} q_{ik}(W_i^\top W_k)(W_k^\top A_i^*)A_k^* + \sum_{\substack{j,k=1\\ j\neq k\neq i}}^{h} q_{ijk}(W_i^\top W_j)(W_j^\top A_i^*)A_k^*\\
&+ \sum_{\substack{k,l=1\\ l\neq k\neq i}}^{h} q_{ikl}(W_i^\top W_i)(W_i^\top A_l^*)A_k^* + \sum_{\substack{k,l=1\\ l\neq k\neq i}}^{h} q_{ikl}(W_i^\top W_k)(W_k^\top A_l^*)A_k^* + \sum_{\substack{k,l=1\\ l\neq k\neq i}}^{h} q_{ikl}(W_i^\top W_l)(W_l^\top A_l^*)A_k^*\\
&+ \sum_{\substack{j,k,l=1\\ j\neq k\neq l\neq i}}^{h} q_{ijkl}(W_i^\top W_j)(W_j^\top A_l^*)A_k^*\Bigg\}
\end{aligned}
$$

We plugin $\epsilon_i = 2m_1h^p\left(\delta + \frac{\mu}{\sqrt{n}}\right)$ for $i = 1,\dots,h$ in the above to get,

$$
\begin{aligned}
||\hat{e_{i4}}|| &\leq 2m_1^2h^{3p-1}\left(\delta+\frac{\mu}{\sqrt{n}}\right)^2+2m_1^2h^{4p-1}\left(\delta+\frac{\mu}{\sqrt{n}}\right)\left(\delta^2+2\delta+\frac{\mu}{\sqrt{n}}\right)\\
&+2m_1^2h^{3p-1}\left(\delta+\frac{\mu}{\sqrt{n}}\right)(1+\delta)^2\\
&+m_1^2||\mathbf{b}||+m_1^2h^{2p-1}\left(\delta+\frac{\mu}{\sqrt{n}}\right)\left(\delta^2+2\delta+\frac{\mu}{\sqrt{n}}\right)+m_1^2h^{3p-1}\left(\delta+\frac{\mu}{\sqrt{n}}\right)\left(\delta^2+2\delta+\frac{\mu}{\sqrt{n}}\right)\\
&+m_1^2h^{3p-1}(1+\delta)^2\left(\delta+\frac{\mu}{\sqrt{n}}\right)+m_1^2h^{3p-1}\left(\delta+\frac{\mu}{\sqrt{n}}\right)\left(\delta^2+2\delta+\frac{\mu}{\sqrt{n}}\right)\\
&+m_1^2h^{3p-1}(1+\delta)\left(\delta^2+2\delta+\frac{\mu}{\sqrt{n}}\right)\\
&+m_1^2h^{4p-1}\left(\delta+\frac{\mu}{\sqrt{n}}\right)\left(\delta^2+2\delta+\frac{\mu}{\sqrt{n}}\right)\\
\implies ||\hat{e_{i4}}|| &\leq 2m_1^2h^{3p-1}\left(\delta+\frac{\mu}{\sqrt{n}}\right)^2+3m_1^2h^{4p-1}\left(\delta+\frac{\mu}{\sqrt{n}}\right)\left(\delta^2+2\delta+\frac{\mu}{\sqrt{n}}\right)\\
&+3m_1^2h^{3p-1}\left(\delta+\frac{\mu}{\sqrt{n}}\right)(1+\delta)^2\\
&+m_1^2||\mathbf{b}||+m_1^2h^{2p-1}\left(\delta+\frac{\mu}{\sqrt{n}}\right)\left(\delta^2+2\delta+\frac{\mu}{\sqrt{n}}\right)+2m_1^2h^{3p-1}\left(\delta+\frac{\mu}{\sqrt{n}}\right)\left(\delta^2+2\delta+\frac{\mu}{\sqrt{n}}\right)\\
&+m_1^2h^{3p-1}(1+\delta)\left(\delta^2+2\delta+\frac{\mu}{\sqrt{n}}\right)\\
\implies ||\hat{e_{i4}}|| &\leq 2m_1^2h^{3p-1}(h^{-2p-2\nu^2}+2h^{-p-\nu^2-\xi}+h^{-2\xi})\\
&+3m_1^2h^{4p-1}(h^{-3p-3\nu^2}+2h^{-2p-2\nu^2}+3h^{-p-\nu^2-\xi}+h^{-2p-2\nu^2-\xi}+h^{-2\xi})\\
&+3m_1^2h^{3p-1}(h^{-3p-3\nu^2}+2h^{-2p-2\nu^2}+2h^{-p-\nu^2-\xi}+h^{-2p-2\nu^2-\xi}+h^{-\xi}+h^{-p-\nu^2})\\
&+m_1^2||\mathbf{b}||\\
&+m_1^2h^{2p-1}(h^{-3p-3\nu^2}+2h^{-2p-2\nu^2}+3h^{-p-\nu^2-\xi}+h^{-2p-2\nu^2-\xi}+h^{-2\xi})\\
&+2m_1^2h^{3p-1}(h^{-3p-3\nu^2}+2h^{-2p-2\nu^2}+3h^{-p-\nu^2-\xi}+h^{-2p-2\nu^2-\xi}+h^{-2\xi})\\
&+m_1^2h^{3p-1}(h^{-3p-3\nu^2}+3h^{-2p-2\nu^2}+h^{-p-\nu^2-\xi}+h^{-\xi}+2h^{-p-\nu^2})\\
\implies ||\hat{e_{i4}}|| &\leq 2m_1^2h^{p-1}(h^{-2\nu^2}+2h^{-\nu^2+p-\xi}+h^{2p-2\xi})\\
&+3m_1^2h^{p-1}(h^{-3\nu^2}+2h^{-p-2\nu^2}+3h^{p-\nu^2+p-\xi}+h^{-2\nu^2+p-\xi}+h^{3p-2\xi})\\
&+3m_1^2h^{p-1}(h^{-p-3\nu^2}+2h^{-2\nu^2}+2h^{-\nu^2+p-\xi}+h^{-2\nu^2-\xi}+h^{2p-\xi}+h^{p-\nu^2})\\
&+m_1^2||\mathbf{b}||\\
&+m_1^2h^{p-1}(h^{-2p-3\nu^2}+2h^{-p-2\nu^2}+3h^{-\nu^2-\xi}+h^{-p-2\nu^2-\xi}+h^{p-2\xi})\\
&+2m_1^2h^{p-1}(h^{-p-3\nu^2}+2h^{-2\nu^2}+3h^{-\nu^2+p-\xi}+h^{-2\nu^2-\xi}+h^{2p-2\xi})\\
&+m_1^2h^{p-1}(h^{-p-3\nu^2}+3h^{-2\nu^2}+h^{-\nu^2+p-\xi}+h^{2p-\xi}+2h^{p-\nu^2})
\end{aligned}
$$

Now let us find a bound for $||\mathbf{b}||$.

$$\begin{aligned}\mathbf{b} &= \sum_{\substack{k=1\\k\neq i}}^{h} q_{ik}(W_i^\top W_i)(W_i^\top A_i^*)A_k^* \\ &= \langle W_i, W_i\rangle\langle W_i, A_i^*\rangle q_{ik} A_{-i}^* \mathbf{1}_h\end{aligned}$$

Where A^*_{-i} is the dictionary A^* with the ith column set to zero, and $\mathbf{1}_h \in \mathbb{R}^h$ is the h-dimensional vector of all ones. Here we make use of the distributional assumption that q_{ik} is the same for all i, k in order to pull q_{ik} out of the sum.

$$\begin{aligned}||\mathbf{b}||_2 &= h^{2p-2}\langle W_i, W_i\rangle\langle W_i, A_i^*\rangle||A_{-i}^*\mathbf{1}_h||_2 \\ &\leq h^{2p-2}(1+\delta)^3||A_{-i}^*||_2||\mathbf{1}_h||_2 \\ &= h^{2p-2}(1+\delta)^3 h^{1/2}\sqrt{\lambda_{\max}(A_{-i}^{*\top}A_{-i}^*)} \\ &= h^{2p-2}(1+\delta)^3 h^{1/2}\sqrt{h\frac{\mu}{\sqrt{n}}+1} \\ &= h^{p-1}\sqrt{h^{2p-2}\times h\times(1+\delta)^6\times\left(h\frac{\mu}{\sqrt{n}}+1\right)} \\ &= h^{p-1}\sqrt{h^{2p-1}\times(1+h^{-p-\nu^2})^6\times\left(h^{1-\xi}+1\right)} \\ &= h^{p-1}\sqrt{(1+h^{-p-\nu^2})^6\times(h^{2p-\xi}+h^{2p-1})}\end{aligned}$$

Here $||A^*_{-i}||_2$ is the spectral norm of A^*_{-i}, and is the top singular value of the matrix. We use Gershgorin's Circle theorem to bound the top eigenvalue of $A^{*\top}_{-i}A^*_{-i}$ by its maximum row sum.
If $p < \frac{\xi}{2}$, $p < \frac{1}{2}$, and $p < \nu^2$, then $||\hat{e_{i4}}|| = o(m_1^2 h^{p-1})$. Now we combine the above obtained bounds for $\|\hat{e_{it}}\|$ (for $t \in \{1,2,3,4\}$) with the bound obtained below equation 4.7 to say that, $\|e_i\| = o(\max\{m_1^2, m_2\}h^{p-1})$

4.B.3 About $\alpha_i - \beta_i$

Remembering that $D = 1$ and doing a close scrutiny of the terms in 4.6 and 4.4 will indicate that the coefficients are the *same* for the $m_2 h^{p-1}$ term in each of them. (which is the term with the highest h scaling in the m_2 dependent parts of α_i and β_i). So this largest term cancels off in the difference and we are left with the sub-leading order terms coming from both their m_1^2 as well as the m_2 parts and this gives us,

$$\alpha_i - \beta_i = o(\max\{m_1^2, m_2\}h^{p-1})$$

Chapter 5

Understanding Adaptive Gradient Algorithms

5.1 Introduction

Many optimization questions arising in machine learning can be cast as a finite sum optimization problem of the form: $\min_{\mathbf{x}} f(\mathbf{x})$ where $f(\mathbf{x}) = \frac{1}{k}\sum_{i=1}^{k} f_i(\mathbf{x})$. Most neural network problems also fall under a similar structure where each function f_i is typically non-convex. A well-studied algorithm to solve such problems is Stochastic Gradient Descent (SGD), which uses updates of the form: $\mathbf{x}_{t+1} := \mathbf{x}_t - \alpha \nabla f_{i_t}(\mathbf{x}_t)$, where α is a step size, and f_{i_t} is a function chosen randomly from $\{f_1, f_2, \ldots, f_k\}$ at time t.

Often in neural networks, "momentum" is added to the SGD update to yield a two-step update process given as: $\mathbf{v}_{t+1} = \mu \mathbf{v}_t - \alpha \nabla \tilde{f}_{i_t}(\mathbf{x}_t)$ followed by $\mathbf{x}_{t+1} = \mathbf{x}_t + \mathbf{v}_{t+1}$. This algorithm is typically called the Heavy-Ball (HB) method (or sometimes classical momentum), with $\mu > 0$ called the momentum parameter (Polyak, 1987). In the context of neural nets, another variant of SGD that is popular is Nesterov's Accelerated Gradient (NAG), which can also be thought of as a momentum method (Sutskever et al., 2013), and has updates of the form $\mathbf{v}_{t+1} = \mu \mathbf{v}_t - \alpha \nabla \tilde{f}_{i_t}(\mathbf{x}_t + \mu \mathbf{v}_t)$ followed by $\mathbf{x}_{t+1} = \mathbf{x}_t + \mathbf{v}_{t+1}$ (see Algorithm 5 for more details).

Momentum methods like HB and NAG have been shown to have superior convergence properties compared to gradient descent both for convex and non-convex functions (Nesterov, 1983; Polyak, 1987),
(Zavriev and Kostyuk, 1993; Ochs, 2016; O'Neill and Wright, 2017; Jin, Netrapalli, and Jordan, 2017). To the best of our knowledge, when using a stochastic gradient oracle there is no clear theoretical justification yet known of the benefits of NAG and HB over regular SGD in general (Yuan, Ying, and Sayed, 2016; Kidambi et al., 2018; Wiegerinck, Komoda, and Heskes, 1994; Yang, Lin, and Li, 2016; Gadat, Panloup, Saadane, et al., 2018), unless considering specialized function classes (Loizou and Richtárik, 2017). But in practice, these momentum methods, and in particular NAG, have been

repeatedly shown to have good convergence and generalization on a range of neural net problems (Sutskever et al., 2013; Lucas, Zemel, and Grosse, 2018; Kidambi et al., 2018).

The performance of NAG (as well as HB and SGD), however, are typically quite sensitive to the selection of its hyper-parameters: step size, momentum and batch size (Sutskever et al., 2013). Thus, "adaptive gradient" algorithms such as RMSProp (Algorithm 6) (Tieleman and Hinton, 2012) and ADAM (Algorithm 7) (Kingma and Ba, 2014) have become very popular for optimizing deep neural networks (Melis, Dyer, and Blunsom, 2017; Denkowski and Neubig, 2017; Gregor et al., 2015; Radford, Metz, and Chintala, 2015; Bahar et al., 2017). The reason for their widespread popularity seems to be the fact that they are easier to tune than SGD, NAG or HB. Adaptive gradient methods use as their update direction a vector which is the image of a linear combination of all the gradients seen till now, under a linear transformation (often called the "diagonal pre-conditioner") constructed out of the history of the gradients. It is generally believed that this "pre-conditioning" makes these algorithms much less sensitive to the selection of its hyper-parameters. A precursor to RMSProp and ADAM was the AdaGrad algorithm, (Duchi, Hazan, and Singer, 2011).

Despite their widespread use in the deep-learning community, till our work, adaptive gradients methods like RMSProp and ADAM have lacked any theoretical justifications in the non-convex setting - even with exact/deterministic gradients (Bernstein et al., 2018). On the contrary, intriguing recent works like Wilson et al., 2017 and Keskar and Socher, 2017 have shown cases where SGD (no momentum) and HB (classical momentum) generalize much better than RMSProp and ADAM with stochastic gradients. In particular Wilson et al., 2017 showed that ADAM generalizes poorly for large enough nets and that RMSProp generalizes better than ADAM on a couple of neural network tasks (most notably in the character-level language modeling task). But in general it's not clear and no heuristics are known to the best of our knowledge to decide whether these insights about relative performances (generalization or training) between algorithms hold for other models or carry over to the full-batch setting.

Most notably in Reddi, Kale, and Kumar, 2018 the authors showed that in the setting of online convex optimization there are certain sequences of convex functions where ADAM and RMSprop fail to converge to asymptotically zero average regret.

5.1.1 A summary of our contributions

In this work we shed light on the above described open questions about adaptive gradient methods in the following two ways.

- To the best of our knowledge, this work gives the first convergence guarantees for RMSProp and ADAM under any setting. Specifically (a) in Section 5.3 we show stochastic gradient oracle conditions for which RMSProp can converge to approximate criticality for smooth non-convex objectives. Most interesting among these is the "interpolating" oracle condition that we motivate and which we show helps stochastic RMSPRop converge at gradient descent speeds. (b) In Section 5.4 we show run-time bounds s.t for certain regimes of hyper-parameters and classes of smooth non-convex functions deterministic RMSProp and ADAM can reach approximate criticality.

- Our second contribution (in Section 5.6) is to undertake a detailed empirical investigation into adaptive gradient methods, targeted to probe the competitive advantages of RMSProp and ADAM. We compare the convergence and generalization properties of RMSProp and ADAM against NAG on (a) a variety of autoencoder experiments on MNIST data, in both full and mini-batch settings and (b) on image classification task on CIFAR-10 using a VGG-9 convolutional neural network in the mini-batch setting.

  In the full-batch setting, we demonstrate that ADAM with very high values of the momentum parameter ($\beta_1 = 0.99$) matches or outperforms carefully tuned NAG and RMSProp, in terms of getting lower training and test losses. We show that as the autoencoder size keeps increasing, RMSProp fails to generalize pretty soon. In the mini-batch experiments we see exactly the same behaviour for large enough nets.

  We also demonstrate the enhancement in ADAM's ability to get lower population risk values and gradient norms when the ξ parameter is increased. Thus we conclude that this is a crucial hyperparameter that was incidentally not tuned in studies like Wilson et al., 2017

Remark. The counterexample to ADAM's convergence constructed in Theorem 3 in Reddi, Kale, and Kumar, 2018 is in the stochastic optimization framework and is incomparable to our result about deterministic ADAM. Thus our result establishes a key conceptual point that for adaptive gradient algorithms one cannot transfer intuitions about convergence from online setups to their more common use case in offline setups.

On the experimental side we note that recently it has been shown by Lucas, Zemel, and Grosse, 2018, that there are problems where NAG generalizes better than ADAM even after tuning β_1 (see Algorithm 7). In contrast our experiments reveal controlled setups where tuning ADAM's β_1 closer to 1 than usual practice helps close the generalization gap with NAG and HB which exists at standard values of β_1.

5.1.2 Comparison with concurrent proofs in literature

Much after this work was completed we came to know of Li and Orabona, 2018 and Ward, Wu, and Bottou, 2019 which analyzed similar questions as us though none of them address RMSProp or ADAM. The latter of these two shows convergence on smooth non-convex objectives of a form of AdaGrad where adaptivity is limited to only rescaling the currently sampled stochastic gradient. In a similar setup the former reference analyzes convergence rates of a modification of AdaGrad where the currently sampled stochastic gradient does not affect the pre-conditioner. We emphasize that this is a conceptually significant departure from the framework of famously successful adaptive gradient algorithms and experimentally this modification can be shown to hurt the performance. After the initial version of our work De, Mukherjee, and Ullah, 2018 was made public, a flurry of activity happened in this field towards trying to prove better convergence results for ADAM and RMSProp like algorithms, (Chen et al., 2018; Zhou et al., 2018a; Zou et al., 2018b; Zaheer et al., 2018) and (Chen and Gu, 2018). Most recently in Staib et al., 2019 a massive modification of RMSProp has been shown to have the ability to converge to approximate second order critical points.

For the convergence proofs to work the above papers have introduced one or more of the following modifications : (1) they introduce time-dependence in the adaptivity parameters, β_1 (that controls the momentum adaptivity) and the β_2 (that controls the historical contribution of the squared gradients) and (2) they introduce many extra steps (like most notably in Staib et al., 2019 and Chen and Gu, 2018) than there are in the standard software implementations of ADAM or RMSProp which are successful in the real world.

Unlike all the above results we do *not* modify the structure of the standard implementations of RMSProp or ADAM and give the first-of-its-kind characterizations of different conditions for their convergence.

5.2 Pseudocodes

Towards stating the pesudocodes used for NAG, RMSProp and ADAM in theory and experiments, we need the following definition of square-root of diagonal matrices,

Definition 17. Square root of the Penrose inverse If $\mathbf{v} \in \mathbb{R}^d$ and $V = \text{diag}(\mathbf{v})$ then we define, $V^{-\frac{1}{2}} := \sum_{i \in \text{Support}(\mathbf{v})} \frac{1}{\sqrt{\mathbf{v}_i}} \mathbf{e}_i \mathbf{e}_i^T$, where $\{\mathbf{e}_i\}_{\{i=1,\ldots,d\}}$ is the standard basis of $\mathbb{R}^d$

Algorithm 5 Nesterov's Accelerated Gradient (NAG)

Input : A step size α, momentum $\mu \in [0,1)$, and an initial starting point $\mathbf{x}_1 \in \mathbb{R}^d$, and we are given query access to a (possibly noisy) oracle for gradients of $f : \mathbb{R}^d \to \mathbb{R}$.
function NAG($\mathbf{x}_1, \alpha, \mu$)
  Initialize : $\mathbf{v}_1 = \mathbf{0}$
  for $t = 1, 2, \ldots$ do
    $\mathbf{v}_{t+1} = \mu \mathbf{v}_t + \nabla f(\mathbf{x}_t)$
    $\mathbf{x}_{t+1} = \mathbf{x}_t - \alpha(\nabla f(\mathbf{x}_t) + \mu \mathbf{v}_{t+1})$
  end for
end function

Algorithm 6 RMSProp

Input : A constant vector $\mathbb{R}^d \ni \xi \mathbf{1}_d \geq 0$, parameter $\beta_2 \in [0,1)$, step size α, initial starting point $\mathbf{x}_1 \in \mathbb{R}^d$, and we are given query access to a (possibly noisy) oracle for gradients of $f : \mathbb{R}^d \to \mathbb{R}$.
function RMSProp($\mathbf{x}_1, \beta_2, \alpha, \xi$)
  Initialize : $\mathbf{v}_0 = \mathbf{0}$
  for $t = 1, 2, \ldots$ do
    $\mathbf{g}_t = \nabla f(\mathbf{x}_t)$
    Define $\mathbf{g}_t^2 \in \mathbb{R}^d$ s.t $(\mathbf{g}_t^2)_i = (\mathbf{g}_{t,i})^2, \ \forall i \in \{1, \ldots, d\}$
    $\mathbf{v}_t = \beta_2 \mathbf{v}_{t-1} + (1 - \beta_2)(\mathbf{g}_t^2 + \xi \mathbf{1}_d)$
    $V_t = \text{diag}(\mathbf{v}_t)$
    $\mathbf{x}_{t+1} = \mathbf{x}_t - \alpha V_t^{-\frac{1}{2}} \mathbf{g}_t$
  end for
end function

Algorithm 7 ADAM

Input : A constant vector $\mathbb{R}^d \ni \xi \mathbf{1}_d > 0$, parameters $\beta_1, \beta_2 \in [0,1)$, a sequence of step sizes $\{\alpha_t\}_{t=1,2..}$, initial starting point $\mathbf{x}_1 \in \mathbb{R}^d$, and we are given oracle access to the gradients of $f : \mathbb{R}^d \to \mathbb{R}$.

function ADAM($\mathbf{x}_1, \beta_1, \beta_2, \alpha, \xi$)

  Initialize : $\mathbf{m}_0 = \mathbf{0}$, $\mathbf{v}_0 = \mathbf{0}$

  for $t = 1, 2, \ldots$ do

    $\mathbf{g}_t = \nabla f(\mathbf{x}_t)$

    $\mathbf{m}_t = \beta_1 \mathbf{m}_{t-1} + (1-\beta_1)\mathbf{g}_t$

    Define $\mathbf{g}_t^2 \in \mathbb{R}^d$ s.t $(\mathbf{g}_t^2)_i = (\mathbf{g}_{t,i})^2, \ \forall i \in \{1, \ldots, d\}$

    $\mathbf{v}_t = \beta_2 \mathbf{v}_{t-1} + (1-\beta_2)\mathbf{g}_t^2$

    $V_t = \mathrm{diag}(\mathbf{v}_t)$

    $\mathbf{x}_{t+1} = \mathbf{x}_t - \alpha_t \left(V_t^{\frac{1}{2}} + \mathrm{diag}(\xi \mathbf{1}_d) \right)^{-1} \mathbf{m}_t$

  end for

end function

5.3 Sufficient conditions for convergence to criticality for stochastic RMSProp

Previously it has been shown in Rangamani et al., 2017 that mini-batch RMSProp can off-the-shelf do autoencoding on depth 2 autoencoders trained on MNIST data while similar results using non-adaptive gradient descent methods requires much tuning of the step-size schedule. Here we give the first results about convergence to criticality for stochastic RMSProp. Towards that we need the following definitions,

Definition 18. $L-$**smoothness** If $f : \mathbb{R}^d \to \mathbb{R}$ is at least once differentiable then we call it $L-$smooth for some $L > 0$ if for all $\mathbf{x}, \mathbf{y} \in \mathbb{R}^d$ the following inequality holds, $f(\mathbf{y}) \leq f(\mathbf{x}) + \langle \nabla f(\mathbf{x}), \mathbf{y} - \mathbf{x} \rangle + \frac{L}{2}\|\mathbf{y} - \mathbf{x}\|^2$

Definition 19 ($(\xi, c, f)-$**Interpolating Oracle).**

For some $\xi > 0$ and $c > 0$ and an atleast once differentiable objective function $f : \mathbb{R}^d \to \mathbb{R}$, a $(\xi, c, f)-$*Interpolating Oracle* when queried at $\mathbf{x}_t \in \mathbb{R}^d$ replies with the vector $\mathbf{g}_t \in \mathbb{R}^d$ s.t it satisfies

the following inequality,

$$\mathbb{E}\left[\left(\|\mathbf{g}_t\| + \frac{\sqrt{d\xi}}{2}\right)^2\right] \leq \left(\sqrt{c}\|\nabla f(\mathbf{x}_t)\| - \frac{\sqrt{d\xi}}{2}\right)^2$$

Remark. Seeing the stochastic algorithm as a stochastic process $\{\mathbf{x}_1, \ldots\}$, in the proof we will need the above inequality to hold only for the conditional expectation of $\left(\|\mathbf{g}_t\| + \frac{\sqrt{d\xi}}{2}\right)^2$ w.r.t the the sigma algebra generated by $\{\mathbf{x}_1, \ldots, \mathbf{x}_t\}$.

Intuition for the above oracle condition In a typical use-case of ADAM or RMSProp, $\mathbf{g}_t$ is an unbiased estimate of the gradient of the empirical loss f given as, $f = \frac{1}{k}\sum_{i=1}^{k} f_i$ where f_i is the $\mathbb{R}^d \to \mathbb{R}$ loss function evaluated on the $i^{th}-$data point. If one were training say neural nets then the *"d" above would be the number of trainable parameters of the net* which is typically in tens of millions. When queried at parameter value $\mathbf{x}_t \in \mathbb{R}^d$, a standard instantiation of the oracle is that it returns, $\mathbf{g}_t = \nabla f_i(\mathbf{x}_t)$ after sampling f_i uniformly at random from $\{f_j\}_{j=1}^{k}$. Suppose $\mathbf{x}_c$ is a parameter value s.t it is critical to all the $\{f_j\}_{j=1}^{k}$ then $\mathbf{x}_c$ is also a critical point of f. If the class of functions is large enough (like those corresponding to deep nets used in practice) that for some parameter values it can interpolate the training data and then for loss functions lowerbounded by 0, such candidate $\mathbf{x}_c$s are these interpolating parameter values.

By continuity of the f_is, the above oracle when queried in a neighbourhood of the interpolating $\mathbf{x}'$s returns a vector of infinitesimal norm and in those neighbourhoods the true gradient is also infinitesimal. Thus we can see that the oracle condition proposed in definition 19 is a way to abstractly capture this phenomenon.

Now we can demonstrate the power of this definition by proving the following theorem which leverages this condition gives the first proof of convergence of stochastic RMSProp.

Theorem 5.3.1. Fast Stochastic RMSProp with an "Interpolating Oracle" (Proof in Section 5.7.1) Suppose $f : \mathbb{R}^d \to \mathbb{R}$ is $L-$smooth and $\exists\ \sigma > 0$ s.t $\|\nabla_i f(\mathbf{x})\| \leq \sigma$ for all $\mathbf{x} \in \mathbb{R}^d$ and $i \in \{1, \ldots, d\}$. Now suppose that we run the RMSProp algorithm as defined in Algorithm 6 (with query access to conditionally unbiased estimator of the gradient of f) and the oracle additionally satisfying the $(\xi, c, f)-$overparameterization condition given in Definition 19 s.t $\sigma < \left(\frac{\xi}{2c}\right)^{\frac{2}{3}}$ and β_2 is chosen so that, $\frac{c\sigma^{1.5}}{\xi} < \sqrt{\beta_2(1-\beta_2)}$. [1] Then there exists a choice of constant step-size α for the algorithm such

[1] Since the constants c, σ and ξ are constrained s.t $\frac{c\sigma^{1.5}}{\xi} < \frac{1}{2}$, it follows that a choice of β_2 as required always exists.

that for $T = O(\frac{1}{\epsilon^2})$ steps we have,

$$\mathbb{E}[\min_{i=1,\ldots,T} \|\nabla f(\mathbf{x}_i)\|^2] \leq O(\epsilon^2)$$

□

Note that here we see a stochastic algorithm being able to converge at the same fast speed as is characteristic of SGD on convex functions. This result can be contrasted with corollary 3 in Zaheer et al., 2018 where similar speeds were motivated for RMSProp with mini-batch sizes being unrealistically large i.e as big as the number of steps to required to converge. In our above theorem such a convergence is seen to arise as a more general phenomenon because of a certain control being true on the expected value of the norm of the gradient oracle's reply. Also note that it is not necessary that the "ξ" parameter in the oracle's definition exactly match the "ξ" parameter of the RMSProp. We note the following corollary which follows from the above theorem and can be proven similarly,

Remark. (a) Note that here we see a stochastic algorithm being able to converge at the same fast speed as is characteristic of SGD on convex functions i.e in T iterations the expected value of the minimum gradient norm found is $O\left(\frac{1}{\sqrt{T}}\right)$ **(b)** Long after this work was completed, we became aware of works like Vaswani, Bach, and Schmidt, 2018 where oracle conditions were introduced of the similar kind as above to show enhanced convergence speeds of much simpler algorithms like SGD.

Corollary 5.3.2. The fast convergence of RMSProp as claimed in Theorem 5.3.1 above also holds if the oracle satisfies the overparameterization condition as given in Definition 19 with a dynamic ξ say $\xi_{\mathbf{x}}$ s.t, $\frac{\xi_{\mathbf{x}} - \xi}{\sqrt{\xi_{\mathbf{x}}}} \leq 2\sqrt{\frac{c}{d}}\|\nabla f(\mathbf{x})\|$ (where we recall that ξ is the hyperparameter in Algorithm 6)

Now we demonstrate yet another situation for which stochastic RMSProp can be shown to converge and this time we directly put constraints on the training data to get the convergence instead of using oracle conditions as above. Towards this we need the following definition,

Definition 20 (The sign function). We define the function sign : $\mathbb{R}^d \to \{-1, 1\}^d$ s.t it maps $\mathbf{v} \mapsto (1 \text{ if } \mathbf{v}_i \geq 0 \text{ else } -1)_{i=1,\ldots,d}$.

Theorem 5.3.3 (Standard speed stochastic RMSProp with a sign constrained oracle (Proof in Appendix 5.A)). Let $f : \mathbb{R}^d \to \mathbb{R}$ be $L-$smooth and be of the form $f = \frac{1}{k}\sum_{p=1}^{k} f_p$ s.t. (a) each f_i is at least once differentiable, (b) the gradients are s.t $\forall \mathbf{x} \in \mathbb{R}^d, \forall p, q \in \{1, \ldots, k\}$, $\text{sign}(\nabla f_p(\mathbf{x})) = \text{sign}(\nabla f_q(\mathbf{x}))$, (c) $\sigma_f < \infty$ is an upperbound on the norm of the gradients of f_i and (d) f has a minimizer, i.e., there exists $\mathbf{x}_*$ such that $f(\mathbf{x}_*) = \min_{\mathbf{x}\in\mathbb{R}^d} f(\mathbf{x})$. Let the gradient oracle be s.t when invoked at some $\mathbf{x}_t \in \mathbb{R}^d$ it uniformly at random picks $i_t \sim \{1, 2, .., k\}$ and returns, $\nabla f_{i_t}(\mathbf{x}_t) = \mathbf{g}_t$. Then corresponding to any $\epsilon, \xi > 0$ and a starting point $\mathbf{x}_1$ for Algorithm 6, we can define, $T \leq \frac{1}{\epsilon^4}\left(\frac{2L\sigma_f^2(\sigma_f^2+\xi)(f(\mathbf{x}_1)-f(\mathbf{x}_*))}{(1-\beta_2)\xi}\right)$ s.t. we are guaranteed that the iterates of Algorithm 6 using a constant step-length of, $\alpha = \frac{1}{\sqrt{T}}\sqrt{\frac{2\xi(1-\beta_2)(f(\mathbf{x}_1)-f(\mathbf{x}_*))}{\sigma_f^2 L}}$ will find an $\epsilon-$critical point in at most T steps in the sense that, $\min_{t=1,2\ldots,T} \mathbb{E}[\|\nabla f(\mathbf{x}_t)\|^2] \leq \epsilon^2$. □

Remark. We note that the theorem above continues to hold even if the constraint (b) that we have about the signs of the gradients of the $\{f_p\}_{p=1,\ldots,k}$ *only* holds on the points in $\mathbb{R}^d$ that the stochastic RMSProp visits and its not necessary for the constraint to be true everywhere in the domain. Further we can say in otherwords that this constraint ensures all the options for the gradient that this stochastic oracle has at any point, to lie in the same orthant of $\mathbb{R}^d$ though this orthant itself can change from one iterate of the next. (And the assumption also ensures that if for some coordinate i, $\nabla_i f = 0$ then for all $p \in \{1, \ldots, k\}$, $\nabla_i f_p = 0$)

5.4 Sufficient conditions for convergence to criticality for non-convex deterministic adaptive gradient algorithms

We note that there are important motivations to study the behavior of neural net training algorithms in the deterministic setting because of use cases where the amount of noise is controlled during optimization, either by using larger batches (Martens and Grosse, 2015; De et al., 2017; Babanezhad et al., 2015) or by employing variance-reducing techniques (Johnson and Zhang, 2013; Defazio, Bach, and Lacoste-Julien, 2014). Inspired by these we also investigate the full-batch RMSProp and ADAM in our controlled autoencoder experiments in Section 5.6.3. Towards that we will now demonstrate that such oracle conditions as in the previous section are not necessary to guarantee convergence of the deterministic RMSProp.

Theorem 5.4.1 (Convergence of deterministic RMSProp - the version with standard speeds (Proof in Appendix 5.B)). Let $f : \mathbb{R}^d \to \mathbb{R}$ be $L-$smooth and let $\sigma < \infty$ be an upperbound on the norm of the

gradient of f. Assume also that f has a minimizer, i.e., there exists $\mathbf{x}_*$ such that $f(\mathbf{x}_*) = \min_{\mathbf{x}\in\mathbb{R}^d} f(\mathbf{x})$. Then the following holds for Algorithm 6 with a deterministic gradient oracle:

For any $\epsilon, \xi > 0$, using a constant step length of $\alpha_t = \alpha = \frac{(1-\beta_2)\xi}{L\sqrt{\sigma^2+\xi}}$ for $t = 1, 2, ...$, guarantees that $\|\nabla f(\mathbf{x}_t)\| \leq \epsilon$ for some $t \leq \frac{1}{\epsilon^2} \times \frac{2L(\sigma^2+\xi)(f(\mathbf{x}_1)-f(\mathbf{x}_*))}{(1-\beta_2)\xi}$, where $\mathbf{x}_1$ is the first iterate of the algorithm. □

One might wonder if the ξ parameter introduced in all the algorithms above is necessary to get convergence guarantees for RMSProp. Towards that in the following theorem we show convergence of another variant of deterministic RMSProp which does not use the ξ parameter and instead uses other assumptions on the objective function and step size modulation. But these tweaks to eliminate the need of ξ come at the cost of the convergence rates getting weaker.

Theorem 5.4.2 (Convergence of deterministic RMSProp - the version with no ξ shift (Proof in Appendix 5.C)). Let $f : \mathbb{R}^d \to \mathbb{R}$ be $L-$smooth and let $\sigma < \infty$ be an upperbound on the norm of the gradient of f. Assume also that f has a minimizer, i.e., there exists $\mathbf{x}_*$ such that $f(\mathbf{x}_*) = \min_{\mathbf{x}\in\mathbb{R}^d} f(\mathbf{x})$, and the function f be bounded from above and below by constants B_ℓ and B_u as $B_l \leq f(\mathbf{x}) \leq B_u$ for all $\mathbf{x} \in \mathbb{R}^d$. Then for any $\epsilon > 0$, $\exists T = O(\frac{1}{\epsilon^4})$ s.t. the Algorithm 6 with a deterministic gradient oracle and $\xi = 0$ is guaranteed to reach a t-th iterate s.t. $1 \leq t \leq T$ and $\|\nabla f(\mathbf{x}_t)\| \leq \epsilon$. □

Next we analyze deterministic ADAM albeit in the small β_1 regime. We note that a small β_1 does not cut-off contributions to the update direction from gradients in the arbitrarily far past (which are typically significantly large), and neither does it affect the non-triviality of the pre-conditioner which does not depend on β_1 at all.

Theorem 5.4.3. Deterministic ADAM converges to criticality (Proof in subsection 5.7.2) Let $f : \mathbb{R}^d \to \mathbb{R}$ be $L-$smooth and let $\sigma < \infty$ be an upperbound on the norm of the gradient of f. Assume also that f has a minimizer, i.e., there exists $\mathbf{x}_*$ such that $f(\mathbf{x}_*) = \min_{\mathbf{x}\in\mathbb{R}^d} f(\mathbf{x})$. Then the following holds for Algorithm 7:

For any $\epsilon > 0$, $\beta_1 < \frac{\epsilon}{\epsilon+\sigma}$ and $\xi > \frac{\sigma^2\beta_1}{-\beta_1\sigma+\epsilon(1-\beta_1)}$, there exist step sizes $\alpha_t > 0$, $t = 1, 2, \ldots$ and a natural number T (depending on β_1, ξ) such that $\|\nabla f(\mathbf{x}_t)\| \leq \epsilon$ for some $t \leq T$.

In particular if one sets $\beta_1 = \frac{\epsilon}{\epsilon+2\sigma}$, $\xi = 2\sigma$, and $\alpha_t = \frac{\|\mathbf{g}_t\|^2}{L(1-\beta_1^t)^2}\frac{4\epsilon}{3(\epsilon+2\sigma)^2}$ where $\mathbf{g}_t$ is the gradient of the objective at the t^{th} iterate, then T can be taken to be $\frac{9L\sigma^2}{\epsilon^6}[f(\mathbf{x}_2) - f(\mathbf{x}_*)]$, where $\mathbf{x}_2$ is the second iterate of the algorithm. □

In other words in T iterates the lowest norm of the gradient encountered by deterministic/"full-batch" ADAM for smooth non-convex objectives falls at least as fast as $O\left(\frac{1}{T^{\frac{1}{6}}}\right)$

Our motivations towards the above theorem were primarily rooted in trying to understand the situations where ADAM as an offline optimizer can converge at all (given the negative results about ADAM in the online setting as in Reddi, Kale, and Kumar, 2018). But we point out that it remains open to tighten the analysis of deterministic ADAM and obtain faster rates than what we have shown in the theorem above and also to be able to characterize conditions when stochastic ADAM can converge.

Remark. It is often believed that ADAM gains over RMSProp because of its so-called "bias correction term" which refers to the step length of ADAM having an iteration dependence of the following form, $\sqrt{1-\beta_2^t}/(1-\beta_1^t)$. As a key success of the above theorem, we note that the $1/(1-\beta_1^t)$ term of this "bias correction term" naturally comes out from theory!

5.5 The Experimental setup

For testing the empirical performance of ADAM and RMSProp, we perform experiments on fully connected autoencoders using ReLU activations and shared weights and on CIFAR-10 using VGG-9, a convolutional neural network. The experiment on VGG-9 has been described in subsection 5.6.5.

To the best of our knowledge there have been very few comparisons of ADAM and RMSProp with other methods on a regression setting and that is one of the main gaps in the literature that we aim to fix by our study here. In a way this also builds on our previous work (Rangamani et al., 2017) (Chapter 4) where we had undertaken a theoretical analysis of autoencoders and in their experiments and had found RMSProp to have good reconstruction error for MNIST when used on even just 2 layer ReLU autoencoders.

To keep our experiments as controlled as possible, we make all layers in a network have the same width (which we denote as h). Thus, we fix the dimensions of the weight matrices of the depth $2\ell-1$, $\mathbb{R}^d \to \mathbb{R}^d$ autoencoders (as defined in Chapter 1) as : $W_1 \in \mathbb{R}^{h\times d}$, $W_i \in \mathbb{R}^{h\times h}, i = 2, \ldots, \ell$. This allowed us to study the effect of increasing depth ℓ or width h without having to deal with added confounding factors. For all experiments, we use the standard "Glorot initialization" for the weights (Glorot and Bengio, 2010), where each element in the weight matrix is initialized by sampling from a

uniform distribution with $[-\text{limit}, \text{limit}]$, limit $= \sqrt{6/(\text{fan}_{\text{in}} + \text{fan}_{\text{out}})}$, where fan_{in} denotes the number of input units in the weight matrix, and fan_{out} denotes the number of output units in the weight matrix. All bias vectors were initialized to zero. No regularization was used.

We performed autoencoder experiments on the MNIST dataset for various network sizes (i.e., different values of ℓ and h). We implemented all experiments using TensorFlow (Abadi et al., 2016) using an NVIDIA GeForce GTX 1080 Ti graphics card. We compared the performance of ADAM and RMSProp with Nesterov's Accelerated Gradient (NAG). All experiments were run for 10^5 iterations. We tune over the hyper-parameters for each optimization algorithm using a grid search as described in Appendix 5.D.

To pick the best set of hyper-parameters, we choose the ones corresponding to the lowest loss on the training set at the end of 10^5 iterations. Further, to cut down on the computation time so that we can test a number of different neural net architectures, we crop the MNIST image from 28×28 down to a 22×22 image by removing 3 pixels from each side (almost all of which is whitespace).

Full-batch experiments We are interested in first comparing these algorithms in the full-batch setting. To do this in a computationally feasible way, we consider a subset of the MNIST dataset (we call this: mini-MNIST), which we build by extracting the first 5500 images in the training set and first 1000 images in the test set in MNIST. Thus, the training and testing datasets in mini-MNIST is 10% of the size of the MNIST dataset. Thus the training set in mini-MNIST contains 5500 images, while the test set contains 1000 images. This subset of the dataset is a fairly reasonable approximation of the full MNIST dataset (i.e., contains roughly the same distribution of labels as in the full MNIST dataset), and thus a legitimate dataset to optimize on.

Mini-batch experiments To test if our conclusions on the full-batch case extend to the mini-batch case, we then perform the same experiments in a mini-batch setup where we fix the mini-batch size at 100. For the mini-batch experiment, we consider the full training set of MNIST, instead of the mini-MNIST dataset considered for the full-batch experiments and we also test on CIFAR-10 using VGG-9, a convolutional neural network.

5.6 Experimental Results

5.6.1 RMSProp and ADAM are sensitive to choice of ξ

The ξ parameter is a feature of the default implementations of RMSProp and ADAM such as in TensorFlow. Most interestingly this strictly positive parameter is crucial for our proofs. In this section we present experimental evidence that attempts to clarify that this isn't merely a theoretical artefact but its value indeed has visible effect on the behaviours of these algorithms. We see in Figure 5.6.1 that on increasing the value of this fixed shift parameter ξ, ADAM in particular, is strongly helped towards getting lower gradient norms and lower test losses though it can hurt its ability to get lower training losses. The plots are shown for optimally tuned values for the other hyper-parameters.

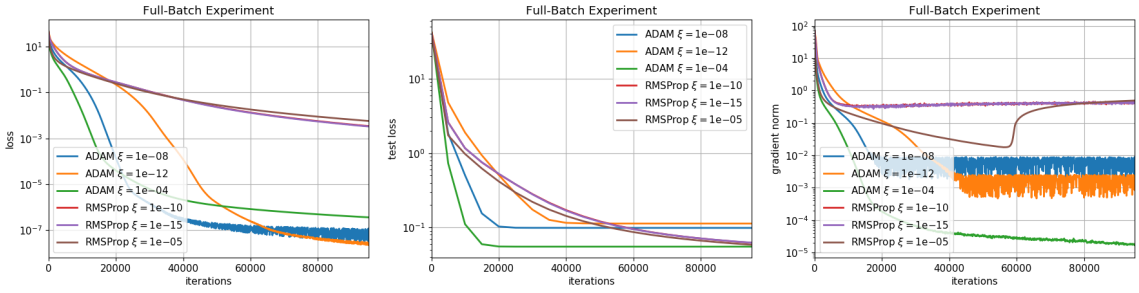


FIGURE 5.6.1: Optimally tuned parameters for different ξ values. 1 hidden layer network of 1000 nodes; *Left*: Loss on training set; *Middle*: Loss on test set; *Right*: Gradient norm on training set

5.6.2 Tracking λ_{min}(Hessian) of the loss function

To check whether NAG, ADAM or RMSProp is capable of consistently moving from a "bad" saddle point to a "good" saddle point region, we track the most negative eigenvalue of the Hessian $\lambda_{\min}(\text{Hessian})$. Even for a very small neural network with around 10^5 parameters, it is still intractable to store the full Hessian matrix in memory to compute the eigenvalues. Instead, we use the Scipy library function `scipy.sparse.linalg.eigsh` that can use a function that computes the matrix-vector products to compute the eigenvalues of the matrix (Lehoucq, Sorensen, and Yang, 1998). Thus, for finding the eigenvalues of the Hessian, it is sufficient to be able to do Hessian-vector products. This can be done exactly in a fairly efficient way (Townsend, 2008).

We display a representative plot in Figure 5.6.2 which shows that NAG in particular has a distinct ability to gradually, but consistently, keep increasing the minimum eigenvalue of the Hessian while continuing to decrease the gradient norm. However unlike as in deeper autoencoders in this case the gradient norms are consistently bigger for NAG, compared to RMSProp and ADAM. In contrast, RSMProp and ADAM quickly get to a high value of the minimum eigenvalue and a small gradient

norm, but somewhat stagnate there. In short, the trend looks better for NAG, but in actual numbers RMSProp and ADAM do better.

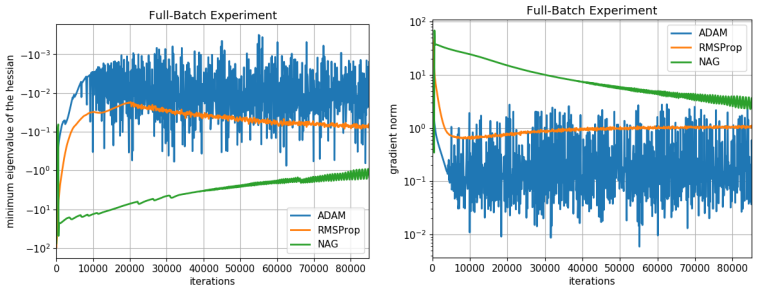


FIGURE 5.6.2: Tracking the smallest eigenvalue of the Hessian on a 1 hidden layer network of size 300. *Left*: Minimum Hessian eigenvalue. *Right*: Gradient norm on training set.

5.6.3 Comparing performance in the full-batch setting

In Figure 5.6.3, we show how the training loss, test loss and gradient norms vary through the iterations for RMSProp, ADAM (at $\beta_1 = 0.9$ and 0.99) and NAG (at $\mu = 0.9$ and 0.99) on a 3 hidden layer autoencoder with 1000 nodes in each hidden layer trained on mini-MNIST. Appendix 5.F.1 and 5.F.2 have more such comparisons for various neural net architectures with varying depth and width and input image sizes, where the following qualitative results also extend.

Conclusions from the full-batch experiments of training autoencoders on mini-MNIST

- Pushing β_1 closer to 1 significantly helps ADAM in getting lower training and test losses and at these values of β_1, it has better performance on these metrics than all the other algorithms. One sees cases like the one displayed in Figure 5.6.3 where ADAM at $\beta_1 = 0.9$ was getting comparable or slightly worse test and training errors than NAG. But once β_1 gets closer to 1, ADAM's performance sharply improves and gets better than other algorithms.

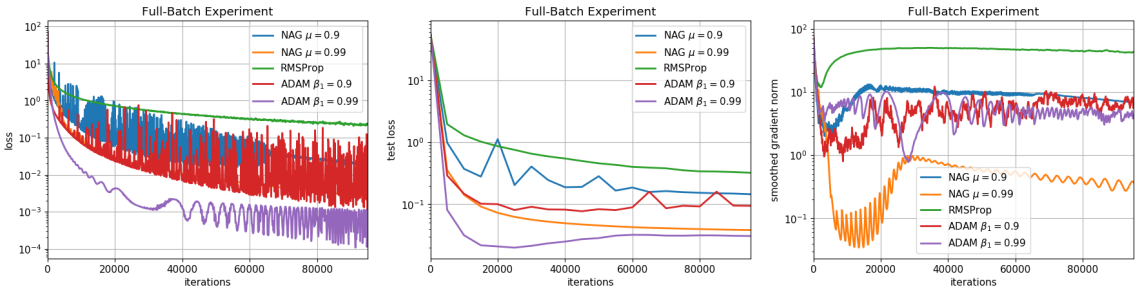


FIGURE 5.6.3: Full-batch experiments on a 3 hidden layer network with 1000 nodes in each layer; *Left*: Loss on training set; *Middle*: Loss on test set; *Right*: Gradient norm on training set

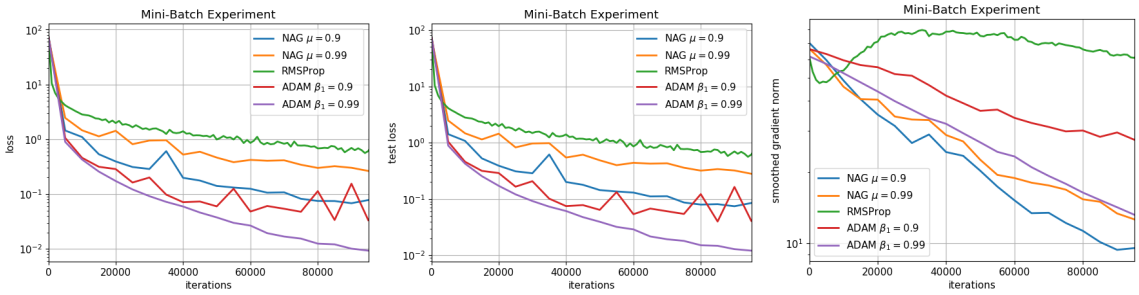


FIGURE 5.6.4: Mini-batch experiments on a network with 5 hidden layers of 1000 nodes each; *Left*: Loss on training set; *Middle*: Loss on test set; *Right*: Gradient norm on training set

- Increasing momentum helps NAG get lower gradient norms though on larger nets it might hurt its training or test performance. NAG does seem to get the lowest gradient norms compared to the other algorithms, except for single hidden layer networks like in Figure 5.6.2.

5.6.4 Corroborating the full-batch behaviors in the mini-batch setting

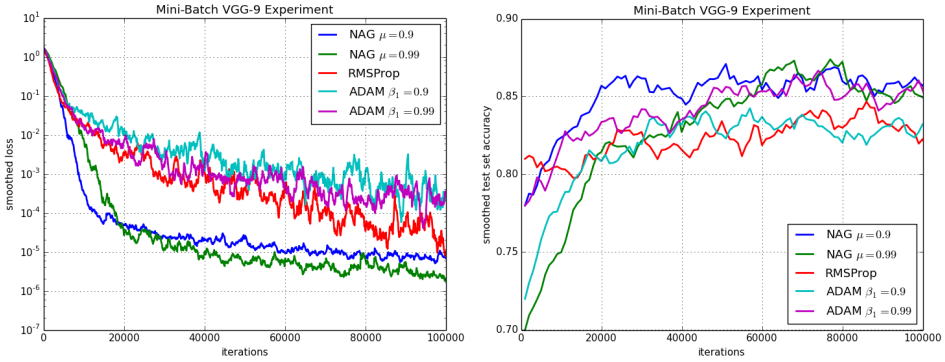
In Figure 5.6.4, we show how training loss, test loss and gradient norms vary when using mini-batches of size 100, on a 5 hidden layer autoencoder with 1000 nodes in each hidden layer trained on the full MNIST dataset. The same phenomenon as here has been demonstrated in more such mini-batch comparisons on autoencoder architectures with varying depths and widths in Appendix 5.F.3 and on VGG-9 with CIFAR-10 in the next subsection 5.6.5.

Conclusions from the mini-batch experiments of training autoencoders on the full MNIST dataset:

- Mini-batching does seem to help NAG do better than ADAM on small nets. However, for larger nets, the full-batch behavior continues, i.e., when ADAM's momentum parameter β_1 is pushed closer to 1, it gets better generalization (significantly lower test losses) than NAG at any momentum tested.

- In general, for all metrics (test loss, training loss and gradient norm reduction) both ADAM as well as NAG seem to improve in performance when their momentum parameter (μ for NAG and β_1 for ADAM) is pushed closer to 1. This effect, which was present in the full-batch setting, seems to get more pronounced here.

- As in the full-batch experiments, NAG continues to have the best ability to reduce gradient norms while for larger enough nets, ADAM at large momentum continues to have the best training error.



(A) Training loss (B) Test set accuracy

FIGURE 5.6.5: Mini-batch image classification experiments with CIFAR-10 using VGG-9

5.6.5 Image Classification on Convolutional Neural Nets

To test whether these results might qualitatively hold for other datasets and models, we train an image classifier on CIFAR-10 (containing 10 classes) using VGG-like convolutional neural networks (Simonyan and Zisserman, 2014). In particular, we train VGG-9 on CIFAR-10, which contains 7 convolutional layers and 2 fully connected layers, a total of 9 layers. The convolutional layers contain 64, 64, 128, 128, 256, 256, 256 filters each of size 3×3, respectively. We use batch normalization (Ioffe and Szegedy, 2015) and ReLU activations after each convolutional layer, and the first fully connected layer. Table 5.6.1 contains more details of the VGG-9 architecture. We use minibatches of size 100, and weight decay of 10^{-5}. We use fixed step sizes, and all hyperparameters were tuned as indicated in Section 5.D.

We present results in Figure 5.6.5. As before, we see that this task is another example where tuning the momentum parameter (β_1) of ADAM helps. While attaining approximately the same loss value, ADAM with $\beta_1 = 0.99$ generalizes as good as NAG and better than when $\beta_1 = 0.9$. Thus tuning β_1 of ADAM helped in closing the generalization gap with NAG.

TABLE 5.6.1: VGG-9 on CIFAR-10.

layer type	kernel size	input size	output size
Conv_1	3×3	$3 \times 32 \times 32$	$64 \times 32 \times 32$
Conv_2	3×3	$64 \times 32 \times 32$	$64 \times 32 \times 32$
Max Pooling	2×2	$64 \times 32 \times 32$	$64 \times 16 \times 16$
Conv_3	3×3	$64 \times 16 \times 16$	$128 \times 16 \times 16$
Conv_4	3×3	$128 \times 16 \times 16$	$128 \times 16 \times 16$
Max Pooling	2×2	$128 \times 16 \times 16$	$128 \times 8 \times 8$
Conv_5	3×3	$128 \times 8 \times 8$	$256 \times 8 \times 8$
Conv_6	3×3	$256 \times 8 \times 8$	$256 \times 8 \times 8$
Conv_7	3×3	$256 \times 8 \times 8$	$256 \times 8 \times 8$
Max Pooling	2×2	$256 \times 8 \times 8$	$256 \times 4 \times 4$
Linear	1×1	1×4096	1×256
Linear	1×1	1×256	1×10

5.7 Proofs of convergence of (stochastic) RMSProp and ADAM

5.7.1 Fast convergence of stochastic RMSProp with "Over Parameterization" (Proof of Theorem 5.3.1)

Proof. By $L-$smoothness of the objective we have the following relationship between the values at consecutive updates,

$$f(\mathbf{x}_{t+1}) \leq f(\mathbf{x}_t) + \langle \nabla f(\mathbf{x}_t), \mathbf{x}_{t+1} - \mathbf{x}_t \rangle + \frac{L}{2}\|\mathbf{x}_{t+1} - \mathbf{x}_t\|^2 \tag{5.1}$$

$$\leq f(\mathbf{x}_t) + \sum_{i=1}^{d} \nabla_i f(\mathbf{x}_t)(\mathbf{x}_{t+1} - \mathbf{x}_t)_i + \frac{L}{2}\sum_{i=1}^{d}(\mathbf{x}_{t+1} - \mathbf{x}_t)_i^2 \tag{5.2}$$

In the last step above we substitute the update rule for the $i^{th}-$coordinate of $\mathbf{x}_t$ as, $\mathbf{x}_{t+1,i} = \mathbf{x}_{t,i} - \frac{\alpha_t \mathbf{g}_{t,i}}{\sqrt{\mathbf{v}_{t,i}}}$ to get,

$$\begin{aligned} f(\mathbf{x}_{t+1}) &\leq f(\mathbf{x}_t) - \alpha_t \sum_{i=1}^{d} \nabla_i f(\mathbf{x}_t) \frac{\mathbf{g}_{t,i}}{\sqrt{\mathbf{v}_{t,i}}} + \frac{L\alpha_t^2}{2}\sum_{i=1}^{d}\frac{\mathbf{g}_{t,i}^2}{(\sqrt{\mathbf{v}_{t,i}})^2} \\ &\leq f(\mathbf{x}_t) - \alpha_t \sum_{i=1}^{d} \nabla_i f(\mathbf{x}_t) \left(\frac{\mathbf{g}_{t,i}}{\sqrt{\mathbf{v}_{t,i}}} - \frac{\mathbf{g}_{t,i}}{\sqrt{\beta_2 \mathbf{v}_{t-1,i}}} + \frac{\mathbf{g}_{t,i}}{\sqrt{\beta_2 \mathbf{v}_{t-1,i}}} \right) \\ &+ \frac{L\alpha_t^2}{2}\sum_{i=1}^{d}\frac{\mathbf{g}_{t,i}^2}{\mathbf{v}_{t,i}} \end{aligned}$$

Now recall that $\mathbb{E}[\mathbf{g}_t \mid \{\mathbf{x}_i\}_{i=1,\ldots,t}] = \nabla f(\mathbf{x}_t)$. We substitute this in the above to get,

$$
\begin{aligned}
&\mathbb{E}[f(\mathbf{x}_{t+1}) \mid \{\mathbf{x}_i\}_{i=1,\dots,t}] \\
&\leq f(\mathbf{x}_t) - \alpha_t \sum_{i=1}^{d} \nabla_i f(\mathbf{x}_t) \Big(\frac{\nabla_i f(\mathbf{x}_t)}{\sqrt{\beta_2 \mathbf{v}_{t-1,i}}} + \mathbb{E}\left[\frac{\mathbf{g}_{t,i}}{\sqrt{\mathbf{v}_{t,i}}} - \frac{\mathbf{g}_{t,i}}{\sqrt{\beta_2 \mathbf{v}_{t-1,i}}} \mid \{\mathbf{x}_i\}_{i=1,\dots,t} \right] \Big) \qquad (5.3) \\
&+ \frac{L\alpha_t^2}{2} \sum_{i=1}^{d} \mathbb{E}\left[\frac{\mathbf{g}_{t,i}^2}{\mathbf{v}_{t,i}} \mid \{\mathbf{x}_i\}_{i=1,\dots,t} \right] \\
&\leq f(\mathbf{x}_t) - \alpha_t \sum_{i=1}^{d} \frac{(\nabla_i f(\mathbf{x}_t))^2}{\sqrt{\beta_2 \mathbf{v}_{t-1,i}}} \\
&+ \alpha_t \sum_{i=1}^{d} |\nabla_i f(\mathbf{x}_t)| \left| \mathbb{E}\left[\frac{\mathbf{g}_{t,i}}{\sqrt{\mathbf{v}_{t,i}}} - \frac{\mathbf{g}_{t,i}}{\sqrt{\beta_2 \mathbf{v}_{t-1,i}}} \mid \{\mathbf{x}_i\}_{i=1,\dots,t} \right] \right| \\
&+ \frac{L\alpha_t^2}{2} \sum_{i=1}^{d} \mathbb{E}\left[\frac{\mathbf{g}_{t,i}^2}{\mathbf{v}_{t,i}} \mid \{\mathbf{x}_i\}_{i=1,\dots,t} \right] \qquad (5.4)
\end{aligned}
$$

Now observe that,

$$
\begin{aligned}
\frac{\mathbf{g}_{t,i}}{\sqrt{\mathbf{v}_{t,i}}} - \frac{\mathbf{g}_{t,i}}{\sqrt{\beta_2 \mathbf{v}_{t-1,i}}} &\leq |\mathbf{g}_{t,i}| \left| \frac{1}{\sqrt{\mathbf{v}_{t,i}}} - \frac{1}{\sqrt{\beta_2 \mathbf{v}_{t-1,i}}} \right| \\
&\leq \left| \frac{\mathbf{g}_{t,i}}{\sqrt{\mathbf{v}_{t,i}}\sqrt{\beta_2 \mathbf{v}_{t-1,i}}} \right| |\sqrt{\beta_2 \mathbf{v}_{t-1,i}} - \sqrt{\mathbf{v}_{t,i}}| \\
&\leq \left| \frac{\mathbf{g}_{t,i}}{\sqrt{\mathbf{v}_{t,i}}\sqrt{\beta_2 \mathbf{v}_{t-1,i}}} \right| \left| \frac{\beta_2 \mathbf{v}_{t-1,i} - \mathbf{v}_{t,i}}{\sqrt{\beta_2 \mathbf{v}_{t-1,i}} + \sqrt{\mathbf{v}_{t,i}}} \right|
\end{aligned}
$$

From the algorithm we have, $\mathbf{v}_{t,i} = \beta_2 \mathbf{v}_{t-1,i} + (1-\beta_2)(\mathbf{g}_{t,i}^2 + \xi)$. We substitute this into the numerator and the denominator of the second factor of the RHS above to get,

$$
\begin{aligned}
&\frac{\mathbf{g}_{t,i}}{\sqrt{\mathbf{v}_{t,i}}} - \frac{\mathbf{g}_{t,i}}{\sqrt{\beta_2 \mathbf{v}_{t-1,i}}} \\
&\leq \left| \frac{\mathbf{g}_{t,i}}{\sqrt{\mathbf{v}_{t,i}}\sqrt{\beta_2 \mathbf{v}_{t-1,i}}} \right| \left| \frac{(1-\beta_2)(\mathbf{g}_{t,i}^2 + \xi)}{\sqrt{\beta_2 \mathbf{v}_{t-1,i}} + \sqrt{\beta_2 \mathbf{v}_{t-1,i} + (1-\beta_2)(\mathbf{g}_{t,i}^2 + \xi)}} \right| \\
&\leq \left| \frac{\mathbf{g}_{t,i}}{\sqrt{\mathbf{v}_{t,i}}\sqrt{\beta_2 \mathbf{v}_{t-1,i}}} \right| \left| \frac{(1-\beta_2)(\mathbf{g}_{t,i}^2 + \xi)}{\sqrt{(1-\beta_2)(\mathbf{g}_{t,i}^2 + \xi)}} \right| = \sqrt{1-\beta_2} \left| \frac{\mathbf{g}_{t,i}\sqrt{\mathbf{g}_{t,i}^2 + \xi}}{\sqrt{\mathbf{v}_{t,i}}\sqrt{\beta_2 \mathbf{v}_{t-1,i}}} \right|
\end{aligned}
$$

Now we substitute the above into equation 5.3 to get,

$$\begin{aligned}
&\mathbb{E}[f(\mathbf{x}_{t+1}) \mid \{\mathbf{x}_i\}_{i=1,\ldots,t}] \\
&\leq f(\mathbf{x}_t) - \alpha_t \sum_{i=1}^{d} \frac{(\nabla_i f(\mathbf{x}_t))^2}{\sqrt{\beta_2 \mathbf{v}_{t-1,i}}} \qquad (5.5) \\
&+ \alpha_t \sqrt{1-\beta_2} \sum_{i=1}^{d} |\nabla_i f(\mathbf{x}_t)| \mathbb{E}\left[\left| \frac{\mathbf{g}_{t,i} \sqrt{\mathbf{g}_{t,i}^2 + \xi}}{\sqrt{\mathbf{v}_{t,i}} \sqrt{\beta_2 \mathbf{v}_{t-1,i}}} \right| \mid \{\mathbf{x}_i\}_{i=1,\ldots,t} \right] \\
&+ \frac{L\alpha_t^2}{2} \sum_{i=1}^{d} \mathbb{E}\left[\frac{\mathbf{g}_{t,i}^2}{\mathbf{v}_{t,i}} \mid \{\mathbf{x}_i\}_{i=1,\ldots,t} \right] \qquad (5.6)
\end{aligned}$$

Now by definition we have, $\mathbf{v}_{t,i} \geq \beta_2 \mathbf{v}_{t-1,i}$ and the definition of σ we infer from the above,

$$\begin{aligned}
\mathbb{E}[f(\mathbf{x}_{t+1}) \mid \{\mathbf{x}_i\}_{i=1,\ldots,t}] &\leq f(\mathbf{x}_t) - \alpha_t \sum_{i=1}^{d} \frac{(\nabla_i f(\mathbf{x}_t))^2}{\sqrt{\beta_2 \mathbf{v}_{t-1,i}}} \\
&+ \sigma \alpha_t \sqrt{1-\beta_2} \sum_{i=1}^{d} \mathbb{E}\left[\left| \frac{\mathbf{g}_{t,i} \sqrt{\mathbf{g}_{t,i}^2 + \xi}}{\beta_2 \mathbf{v}_{t-1,i}} \right| \mid \{\mathbf{x}_i\}_{i=1,\ldots,t} \right] \qquad (5.7) \\
&+ \frac{L\alpha_t^2}{2} \sum_{i=1}^{d} \mathbb{E}\left[\frac{\mathbf{g}_{t,i}^2}{\beta_2 \mathbf{v}_{t-1,i}} \mid \{\mathbf{x}_i\}_{i=1,\ldots,t} \right] \qquad (5.8)
\end{aligned}$$

We have, $\mathbf{v}_t = (1-\beta_2) \sum_{k=1}^{t} \beta_2^{t-k} (\mathbf{g}_k^2 + \xi)$ This implies, $\mathbf{v}_{t,i} \geq (1-\beta_2^t)\xi \geq (1-\beta_2)\xi$. The last inequality follows because we have, $\beta_2 \in (0,1)$ and $t \geq 1$ Substituting this in the above (along with the fact that $\mathbf{v}_{t,i} > 0$) we get,

$$
\begin{aligned}
&\mathbb{E}[f(\mathbf{x}_{t+1}) \mid \{\mathbf{x}_j\}_{j=2,\dots,t}] \\
&\leq f(\mathbf{x}_t) + \sum_{i=1}^{d} \Big(-\alpha_t \frac{(\nabla_i f(\mathbf{x}_t))^2}{\sqrt{\beta_2 \mathbf{v}_{t-1,i}}} + \sigma\alpha_t\sqrt{1-\beta_2} \frac{\mathbb{E}\left[\sqrt{\mathbf{g}_{t,i}^4 + \xi|\mathbf{g}_{t,i}|^2} \mid \{\mathbf{x}_j\}_{j=2,\dots,t}\right]}{\beta_2 \mathbf{v}_{t-1,i}} \\
&+ \frac{L\alpha_t^2}{2} \frac{\mathbb{E}\left[\mathbf{g}_{t,i}^2 \mid \{\mathbf{x}_j\}_{j=2,\dots,t}\right]}{\beta_2 \mathbf{v}_{t-1,i}} \Big) \\
&\leq f(\mathbf{x}_t) - \alpha_t \frac{\|\nabla f(\mathbf{x}_t)\|^2}{\sqrt{\beta_2 \sigma}} \\
&+ \sum_{i=1}^{d} \Big(\sigma\alpha_t\sqrt{1-\beta_2} \frac{\mathbb{E}\left[\sqrt{\mathbf{g}_{t,i}^4 + \xi|\mathbf{g}_{t,i}|^2} \mid \{\mathbf{x}_j\}_{j=2,\dots,t}\right]}{\xi\beta_2(1-\beta_2)} + \frac{L\alpha_t^2}{2} \frac{\mathbb{E}\left[\mathbf{g}_{t,i}^2 \mid \{\mathbf{x}_j\}_{j=2,\dots,t}\right]}{\xi\beta_2(1-\beta_2)} \Big) \\
&\leq f(\mathbf{x}_t) - \alpha_t \frac{\|\nabla f(\mathbf{x}_t)\|^2}{\sqrt{\beta_2 \sigma}} \\
&+ \sum_{i=1}^{d} \Big(\sigma\alpha_t\sqrt{1-\beta_2} + \frac{L\alpha_t^2}{2} \Big) \frac{\mathbb{E}\left[\sqrt{\mathbf{g}_{t,i}^4 + \xi|\mathbf{g}_{t,i}|^2} \mid \{\mathbf{x}_j\}_{j=2,\dots,t}\right]}{\xi\beta_2(1-\beta_2)}
\end{aligned} \tag{5.9}
$$

Now we note that,

$$
\begin{aligned}
&\mathbb{E}\left[\sum_{i=1}^{d} \sqrt{\mathbf{g}_{t,i}^4 + \xi|\mathbf{g}_{t,i}|^2} \mid \{\mathbf{x}_j\}_{j=2,\dots,t}\right] \\
&\leq \mathbb{E}\left[\sum_{i=1}^{d} (\mathbf{g}_{t,i}^2 + \sqrt{\xi}|\mathbf{g}_{t,i}|) \mid \{\mathbf{x}_j\}_{j=2,\dots,t}\right] \leq \mathbb{E}\left[\|\mathbf{g}_t\|^2 + \sqrt{d\xi}\|\mathbf{g}_{t,i}\|) \mid \{\mathbf{x}_j\}_{j=2,\dots,t}\right] \\
&\leq \mathbb{E}\left[\left(\|\mathbf{g}_t\| + \frac{\sqrt{d\xi}}{2}\right)^2 - \frac{d\xi}{4} \mid \{\mathbf{x}_j\}_{j=2,\dots,t}\right]
\end{aligned}
$$

Now we invoke the the property of the oracle given in definition 19 to say that,

$$
\begin{aligned}
&\mathbb{E}\left[\sum_{i=1}^{d} \sqrt{\mathbf{g}_{t,i}^4 + \xi|\mathbf{g}_{t,i}|^2} \mid \{\mathbf{x}_j\}_{j=2,\dots,t}\right] \\
&\leq \mathbb{E}\left[\left(\sqrt{c}\|\nabla f(\mathbf{x}_t)\| - \frac{\sqrt{d\xi}}{2}\right)^2 - \frac{d\xi}{4} \mid \{\mathbf{x}_j\}_{j=2,\dots,t}\right] \\
&\leq c\|\nabla f(\mathbf{x}_t)\|^2
\end{aligned}
$$

Thus substituting the above back into equation 5.9 we get,

$$\mathbb{E}[f(\mathbf{x}_{t+1}) \mid \{\mathbf{x}_j\}_{j=2,\dots,t}] \leq f(\mathbf{x}_t) + \sum_{i=1}^{d} \Big\{ -\frac{\alpha_t}{\sqrt{\beta_2 \sigma}} + c \frac{\Big(\sigma \alpha_t \sqrt{1-\beta_2} + \frac{L\alpha_t^2}{2}\Big)}{\xi \beta_2 (1-\beta_2)} \Big\} \|\nabla_i f(\mathbf{x}_t)\|^2 \tag{5.10}$$

Further we make the optimal choice of $\alpha_t = \frac{\xi \beta_2 (1-\beta_2)}{cL} \Big(\frac{1}{\sqrt{\beta_2 \sigma}} - \frac{c\sigma}{\xi \beta_2 \sqrt{1-\beta_2}} \Big)$ (which is positive by assumptions) and we get,

$$\mathbb{E}[f(\mathbf{x}_{t+1}) \mid \{\mathbf{x}_j\}_{j=2,\dots,t}] \leq f(\mathbf{x}_t) - \frac{1}{2cL} \Big(\frac{c\sigma}{\sqrt{\beta_2 \xi}} - \sqrt{\frac{\xi(1-\beta_2)}{\sigma}} \Big)^2 \|\nabla f(\mathbf{x}_t)\|^2$$

Taking expectation and rearranging we get,

$$\mathbb{E}[\min_{t=1,\dots,T} \|\nabla f(\mathbf{x}_t)\|^2] \leq \frac{1}{T} \sum_{t=1}^{T} \mathbb{E}[\|\nabla f(\mathbf{x}_t)\|^2] \leq \frac{f(\mathbf{x}_1) - f_*}{\frac{T}{2cL} \Big(\frac{c\sigma}{\sqrt{\beta_2 \xi}} - \sqrt{\frac{\xi(1-\beta_2)}{\sigma}} \Big)^2}$$

From here the result follows.

□

5.7.2 Proving ADAM (Proof of Theorem 5.4.3)

Proof. Let us assume to the contrary that $\|g_t\| > \epsilon$ for all $t = 1, 2, 3 \dots$. We will show that this assumption will lead to a contradiction. By $L-$smoothness of the objective we have the following relationship between the values at consecutive updates,

$$f(\mathbf{x}_{t+1}) \leq f(\mathbf{x}_t) + \langle \nabla f(\mathbf{x}_t), \mathbf{x}_{t+1} - \mathbf{x}_t \rangle + \frac{L}{2} \|\mathbf{x}_{t+1} - \mathbf{x}_t\|^2$$

Substituting the update rule using a dummy step length $\eta_t > 0$ we have,

$$f(\mathbf{x}_{t+1}) \leq f(\mathbf{x}_t) - \eta_t \langle \nabla f(\mathbf{x}_t), \left(V_t^{\frac{1}{2}} + \text{diag}(\xi \mathbf{1}_d)\right)^{-1} \mathbf{m}_t \rangle + \frac{L\eta_t^2}{2} \| \left(V_t^{\frac{1}{2}} + \text{diag}(\xi \mathbf{1}_d)\right)^{-1} \mathbf{m}_t \|^2 \tag{5.11}$$

$$\implies f(\mathbf{x}_{t+1}) - f(\mathbf{x}_t) \tag{5.12}$$

$$\leq \eta_t \left(-\langle \mathbf{g}_t, \left(V_t^{\frac{1}{2}} + \text{diag}(\xi \mathbf{1}_d)\right)^{-1} \mathbf{m}_t \rangle + \frac{L\eta_t}{2} \| \left(V_t^{\frac{1}{2}} + \text{diag}(\xi \mathbf{1}_d)\right)^{-1} \mathbf{m}_t \|^2 \right) \tag{5.13}$$

The RHS in equation 5.11 above is a quadratic in η_t with two roots: 0 and $\frac{\langle \mathbf{g}_t, \left(V_t^{\frac{1}{2}} + \text{diag}(\xi \mathbf{1}_d)\right)^{-1} \mathbf{m}_t \rangle}{\frac{L}{2} \| \left(V_t^{\frac{1}{2}} + \text{diag}(\xi \mathbf{1}_d)\right)^{-1} \mathbf{m}_t \|^2}$. So the quadratic's minimum value is at the midpoint of this interval, which gives us a candidate $t^{th}-$step length i.e

$$\alpha_t^* := \frac{1}{2} \cdot \frac{\langle \mathbf{g}_t, \left(V_t^{\frac{1}{2}} + \text{diag}(\xi \mathbf{1}_d)\right)^{-1} \mathbf{m}_t \rangle}{\frac{L}{2} \| \left(V_t^{\frac{1}{2}} + \text{diag}(\xi \mathbf{1}_d)\right)^{-1} \mathbf{m}_t \|^2}$$

and the value of the quadratic at this point is $-\frac{1}{4} \cdot \frac{(\langle \mathbf{g}_t, \left(V_t^{\frac{1}{2}} + \text{diag}(\xi \mathbf{1}_d)\right)^{-1} \mathbf{m}_t \rangle)^2}{\frac{L}{2} \| \left(V_t^{\frac{1}{2}} + \text{diag}(\xi \mathbf{1}_d)\right)^{-1} \mathbf{m}_t \|^2}$. That is with step lengths being this α_t^* we have the following guarantee of decrease of function value between consecutive steps,

$$f(\mathbf{x}_{t+1}) - f(\mathbf{x}_t) \leq -\frac{1}{2L} \cdot \frac{(\langle \mathbf{g}_t, \left(V_t^{\frac{1}{2}} + \text{diag}(\xi \mathbf{1}_d)\right)^{-1} \mathbf{m}_t \rangle)^2}{\| \left(V_t^{\frac{1}{2}} + \text{diag}(\xi \mathbf{1}_d)\right)^{-1} \mathbf{m}_t \|^2} \tag{5.14}$$

Now we separately lower bound the numerator and upper bound the denominator of the RHS above.

Upperbound on $\| \left(V_t^{\frac{1}{2}} + \textbf{diag}(\xi \mathbf{1}_d)\right)^{-1} \mathbf{m}_t \|$

We have, $\lambda_{max}\left(\left(V_t^{\frac{1}{2}} + \text{diag}(\xi \mathbf{1}_d)\right)^{-1}\right) \leq \frac{1}{\xi + \min_{i=1..d} \sqrt{(\mathbf{v}_t)_i}}$ Further we note that the recursion of $\mathbf{v}_t$ can be solved as, $\mathbf{v}_t = (1-\beta_2) \sum_{k=1}^{t} \beta_2^{t-k} \mathbf{g}_k^2$. Now we define, $\epsilon_t := \min_{k=1,..,t, i=1,..,d} (\mathbf{g}_k^2)_i$ and this gives us,

$$\lambda_{max}\left(\left(V_t^{\frac{1}{2}} + \text{diag}(\xi \mathbf{1}_d)\right)^{-1}\right) \leq \frac{1}{\xi + \sqrt{(1-\beta_2^t)\epsilon_t}} \tag{5.15}$$

We solve the recursion for $\mathbf{m}_t$ to get, $\mathbf{m}_t = (1-\beta_1)\sum_{k=1}^{t}\beta_1^{t-k}\mathbf{g}_k$. Then by triangle inequality and defining $\sigma_t := \max_{i=1,..,t}\|\nabla f(\mathbf{x}_i)\|$ we have, $\|\mathbf{m}_t\| \leq (1-\beta_1^t)\sigma_t$. Thus combining this estimate of $\|\mathbf{m}_t\|$ with equation 5.15 we have,

$$\|\Big(V_t^{\frac{1}{2}} + \text{diag}(\xi\mathbf{1}_d)\Big)^{-1}\mathbf{m}_t\| \leq \frac{(1-\beta_1^t)\sigma_t}{\xi + \sqrt{\epsilon_t(1-\beta_2^t)}} \leq \frac{(1-\beta_1^t)\sigma_t}{\xi} \tag{5.16}$$

Lowerbound on $\langle \mathbf{g}_t, \Big(V_t^{\frac{1}{2}} + \textbf{diag}(\xi\mathbf{1}_d)\Big)^{-1}\mathbf{m}_t\rangle$

To analyze this we define the following sequence of functions for each $i = 0, 1, 2.., t$

$$Q_i = \langle \mathbf{g}_t, \Big(V_t^{\frac{1}{2}} + \text{diag}(\xi\mathbf{1}_d)\Big)^{-1}\mathbf{m}_i\rangle$$

This gives us the following on substituting the update rule for $\mathbf{m}_t$,

$$\begin{aligned} Q_i - \beta_1 Q_{i-1} &= \langle \mathbf{g}_t, \Big(V_t^{\frac{1}{2}} + \text{diag}(\xi\mathbf{1}_d)\Big)^{-1}(\mathbf{m}_i - \beta_1\mathbf{m}_{i-1})\rangle \\ &= (1-\beta_1)\langle \mathbf{g}_t, \Big(V_t^{\frac{1}{2}} + \text{diag}(\xi\mathbf{1}_d)\Big)^{-1}\mathbf{g}_i\rangle \end{aligned}$$

At $i = t$ we have, $Q_t - \beta_1 Q_{t-1} \geq (1-\beta_1)\|\mathbf{g}_t\|^2\lambda_{min}\Big(\Big(V_t^{\frac{1}{2}} + \text{diag}(\xi\mathbf{1}_d)\Big)^{-1}\Big)$

Lets define, $\sigma_{t-1} := \max_{i=1,..,t-1}\|\nabla f(\mathbf{x}_i)\|$ and this gives us for $i \in \{1, .., t-1\}$,

$$Q_i - \beta_1 Q_{i-1} \geq -(1-\beta_1)\|\mathbf{g}_t\|\sigma_{t-1}\lambda_{max}\Big(\Big(V_t^{\frac{1}{2}} + \text{diag}(\xi\mathbf{1}_d)\Big)^{-1}\Big)$$

We note the following identity,

$$Q_t - \beta_1^t Q_0 = (Q_t - \beta_1 Q_{t-1}) + \beta_1(Q_{t-1} - \beta_1 Q_{t-2}) + \beta_1^2(Q_{t-2} - \beta_1 Q_{t-3}) + .. \\ + \beta_1^{t-1}(Q_1 - \beta_1 Q_0)$$

Now we use the lowerbounds proven on $Q_i - \beta_1 Q_{i-1}$ for $i \in \{1, .., t-1\}$ and $Q_t - \beta_1 Q_{t-1}$ to lowerbound the above sum as,

$$\begin{aligned} Q_t - \beta_1^t Q_0 &\geq (1-\beta_1)\|\mathbf{g}_t\|^2 \lambda_{min}\Big(\Big(V_t^{\frac{1}{2}} + \text{diag}(\xi \mathbf{1}_d)\Big)^{-1}\Big) \\ &\quad - (1-\beta_1)\|\mathbf{g}_t\|\sigma_{t-1}\lambda_{max}\Big(\Big(V_t^{\frac{1}{2}} + \text{diag}(\xi \mathbf{1}_d)\Big)^{-1}\Big)\sum_{j=1}^{t-1}\beta_1^j \\ &\geq (1-\beta_1)\|\mathbf{g}_t\|^2 \lambda_{min}\Big(\Big(V_t^{\frac{1}{2}} + \text{diag}(\xi \mathbf{1}_d)\Big)^{-1}\Big) \\ &\quad - (\beta_1 - \beta_1^t)\|\mathbf{g}_t\|\sigma_{t-1}\lambda_{max}\Big(\Big(V_t^{\frac{1}{2}} + \text{diag}(\xi \mathbf{1}_d)\Big)^{-1}\Big) \end{aligned} \tag{5.17}$$

We can evaluate the following lowerbound,

$$\lambda_{min}\Big(\Big(V_t^{\frac{1}{2}} + \text{diag}(\xi \mathbf{1}_d)\Big)^{-1}\Big) \geq \frac{1}{\xi + \sqrt{\max_{i=1,..,d}(\mathbf{v}_t)_i}}$$

Next we remember that the recursion of $\mathbf{v}_t$ can be solved as, $\mathbf{v}_t = (1-\beta_2)\sum_{k=1}^t \beta_2^{t-k}\mathbf{g}_k^2$ and we define, $\sigma_t := \max_{i=1,..,t}\|\nabla f(\mathbf{x}_i)\|$ to get,

$$\lambda_{min}\Big(\Big(V_t^{\frac{1}{2}} + \text{diag}(\xi \mathbf{1}_d)\Big)^{-1}\Big) \geq \frac{1}{\xi + \sqrt{(1-\beta_2^t)\sigma_t^2}} \tag{5.18}$$

Now we combine the above and equation 5.15 and the known value of $Q_0 = 0$ (from definition and initial conditions) to get from the equation 5.17,

$$
\begin{aligned}
Q_t &\geq -(\beta_1 - \beta_1^t)\|\mathbf{g}_t\|\sigma_{t-1}\frac{1}{\xi + \sqrt{(1-\beta_2^t)\epsilon_t}} \\
&\quad + (1-\beta_1)\|\mathbf{g}_t\|^2\frac{1}{\xi + \sqrt{(1-\beta_2^t)\sigma_t^2}} \\
&\geq \|\mathbf{g}_t\|^2\left(\frac{(1-\beta_1)}{\xi + \sigma\sqrt{(1-\beta_2^t)}} - \frac{(\beta_1 - \beta_1^t)\sigma}{\xi\|\mathbf{g}_t\|}\right) \qquad (5.19)
\end{aligned}
$$

In the above inequalities we have set $\epsilon_t = 0$ and we have set, $\sigma_t = \sigma_{t-1} = \sigma$. Now we examine the following part of the lowerbound proven above,

$$
\begin{aligned}
&\frac{(1-\beta_1)}{\xi + \sqrt{(1-\beta_2^t)\sigma^2}} - \frac{(\beta_1 - \beta_1^t)\sigma}{\xi\|\mathbf{g}_t\|} \\
&= \frac{\xi\|\mathbf{g}_t\|(1-\beta_1) - \sigma(\beta_1 - \beta_1^t)(\xi + \sigma\sqrt{(1-\beta_2^t)})}{\xi\|\mathbf{g}_t\|(\xi + \sigma\sqrt{(1-\beta_2^t)})} \\
&= \sigma(\beta_1 - \beta_1^t)\frac{\xi\left(\frac{\|\mathbf{g}_t\|(1-\beta_1)}{\sigma(\beta_1 - \beta_1^t)} - 1\right) - \sigma\sqrt{(1-\beta_2^t)}}{\xi\|\mathbf{g}_t\|(\xi + \sigma\sqrt{(1-\beta_2^t)})} \\
&= \sigma(\beta_1 - \beta_1^t)\left(\frac{\|\mathbf{g}_t\|(1-\beta_1)}{\sigma(\beta_1 - \beta_1^t)} - 1\right)\frac{\xi - \left(\frac{\sigma\sqrt{(1-\beta_2^t)}}{-1 + \frac{(1-\beta_1)\|\mathbf{g}_t\|}{(\beta_1 - \beta_1^t)\sigma}}\right)}{\xi\|\mathbf{g}_t\|(\xi + \sigma\sqrt{(1-\beta_2^t)})}
\end{aligned}
$$

Now we remember the assumption that we are working under i.e $\|\mathbf{g}_t\| > \epsilon$. Also by definition $0 < \beta_1 < 1$ and hence we have $0 < \beta_1 - \beta_1^t < \beta_1$. This implies, $\frac{(1-\beta_1)\|\mathbf{g}_t\|}{(\beta_1 - \beta_1^t)\sigma} > \frac{(1-\beta_1)\epsilon}{\beta_1\sigma} > 1$ where the last inequality follows because of our choice of ϵ as stated in the theorem statement. This allows us to define a constant, $\frac{\epsilon(1-\beta_1)}{\beta_1\sigma} - 1 := \theta_1 > 0$ s.t $\frac{(1-\beta_1)\|\mathbf{g}_t\|}{(\beta_1 - \beta_1^t)\sigma} - 1 > \theta_1$ Similarly our definition of ξ allows us to define a constant $\theta_2 > 0$ to get,

$$
\left(\frac{\sigma\sqrt{(1-\beta_2^t)}}{-1 + \frac{(1-\beta_1)\|\mathbf{g}_t\|}{(\beta_1 - \beta_1^t)\sigma}}\right) < \frac{\sigma}{\theta_1} = \xi - \theta_2
$$

Putting the above back into the lowerbound for Q_t in equation 5.19 we have,

$$Q_t \geq \|\mathbf{g}_t\|^2 \left(\frac{\sigma(\beta_1 - \beta_1^2)\theta_1\theta_2}{\xi\sigma(\xi + \sigma)} \right) \tag{5.20}$$

Now we substitute the above and equation 5.16 into equation 5.14 to get,

$$\begin{aligned} f(\mathbf{x}_{t+1}) - f(\mathbf{x}_t) &\leq -\frac{1}{2L} \cdot \frac{(\langle \mathbf{g}_t, \left(V_t^{\frac{1}{2}} + \text{diag}(\xi \mathbf{1}_d)\right)^{-1} \mathbf{m}_t \rangle)^2}{\| \left(V_t^{\frac{1}{2}} + \text{diag}(\xi \mathbf{1}_d)\right)^{-1} \mathbf{m}_t \|^2} \\ &\leq -\frac{1}{2L} \frac{Q_t^2}{\| \left(V_t^{\frac{1}{2}} + \text{diag}(\xi \mathbf{1}_d)\right)^{-1} \mathbf{m}_t \|^2} \\ &\leq -\frac{1}{2L} \frac{\|\mathbf{g}_t\|^4 \left(\frac{(\beta_1 - \beta_1^2)\theta_1\theta_2}{\xi(\xi+\sigma)} \right)^2}{\left(\frac{(1-\beta_1^t)\sigma}{\xi} \right)^2} \\ &\leq -\frac{\|\mathbf{g}_t\|^4}{2L} \left(\frac{(\beta_1 - \beta_1^2)^2 \theta_1^2 \theta_2^2}{(\xi + \sigma)^2 (1 - \beta_1^t)^2 \sigma^2} \right) \end{aligned} \tag{5.21}$$

Thus we get from the above,

$$\begin{aligned} &\left(\frac{(\beta_1 - \beta_1^2)^2 \theta_1^2 \theta_2^2}{2L(\xi + \sigma)^2 (1 - \beta_1^t)^2 \sigma^2} \right) \|\nabla f(\mathbf{x}_t)\|^4 \leq [f(\mathbf{x}_t) - f(\mathbf{x}_{t+1})] \\ &\implies \sum_{t=2}^{T} \left(\frac{(\beta_1 - \beta_1^2)^2 \theta_1^2 \theta_2^2}{2L(\xi + \sigma)^2 \sigma^2} \right) \|\nabla f(\mathbf{x}_t)\|^4 \leq [f(\mathbf{x}_2) - f(\mathbf{x}_{T+1})] \\ &\implies \min_{t=2,..,T} \|\nabla f(\mathbf{x}_t)\|^4 \leq \frac{2L(\xi + \sigma)^2 \sigma^2}{T(\beta_1 - \beta_1^2)^2 \theta_1^2 \theta_2^2} [f(\mathbf{x}_2) - f(\mathbf{x}_*)] \end{aligned}$$

Observe that if, $T \geq \frac{2L\sigma^2(\xi+\sigma)^2}{2\epsilon^4(\beta_1-\beta_1^2)^2\theta_1^2\theta_2^2}[f(\mathbf{x}_2) - f(\mathbf{x}_*)]$ then the RHS of the inequality above is less than or equal to ϵ^4 and this would contradict the assumption that $\|\nabla f(\mathbf{x}_t)\| > \epsilon$ for all $t = 1, 2, \ldots$.

As a consequence we have proven the first part of the theorem which guarantees the existence of positive step lengths, α_t s.t ADAM finds an approximately critical point in finite time.

Now choose $\theta_1 = 1$ i.e $\frac{\epsilon}{2} = \frac{\beta_1\sigma}{1-\beta_1}$ i.e $\beta_1 = \frac{\epsilon}{\epsilon+2\sigma} \implies \beta_1(1-\beta_1) = \frac{\epsilon}{\epsilon+2\sigma}(1 - \frac{\epsilon}{\epsilon+2\sigma}) = \frac{2\sigma\epsilon}{(\epsilon+2\sigma)^2}$. This also gives a easier-to-read condition on ξ in terms of these parameters i.e $\xi > \sigma$. Now choose $\xi = 2\sigma$ i.e $\theta_2 = \sigma$ and making these substitutions gives us,

$$T \geq \frac{18L\sigma^4}{2\epsilon^4 \left(\frac{2\sigma\epsilon}{(\epsilon+2\sigma)}\right)^2 \sigma^2}[f(\mathbf{x}_2) - f(\mathbf{x}_*)] \geq \frac{18L}{8\epsilon^6 \left(\frac{1}{\epsilon+2\sigma}\right)^2}[f(\mathbf{x}_2) - f(\mathbf{x}_*)]$$
$$\geq \frac{9L\sigma^2}{\epsilon^6}[f(\mathbf{x}_2) - f(\mathbf{x}_*)]$$

We substitute these choices in the step length found earlier to get,

$$\alpha_t^* = \frac{1}{L} \cdot \frac{\langle \mathbf{g}_t, \left(V_t^{\frac{1}{2}} + \text{diag}(\xi \mathbf{1}_d)\right)^{-1} \mathbf{m}_t \rangle}{\| \left(V_t^{\frac{1}{2}} + \text{diag}(\xi \mathbf{1}_d)\right)^{-1} \mathbf{m}_t \|^2} = \frac{1}{L} \cdot \frac{Q_t}{\| \left(V_t^{\frac{1}{2}} + \text{diag}(\xi \mathbf{1}_d)\right)^{-1} \mathbf{m}_t \|^2}$$
$$\geq \frac{1}{L} \frac{\|\mathbf{g}_t\|^2 \left(\frac{\sigma^2(\beta_1 - \beta_1^2)}{\xi\sigma(\xi+\sigma)}\right)}{\left(\frac{(1-\beta_1^t)\sigma}{\xi}\right)^2} = \frac{\|\mathbf{g}_t\|^2}{L(1-\beta_1^t)^2} \frac{4\epsilon}{3(\epsilon+2\sigma)^2} := \alpha_t$$

In the theorem statement we choose to call as the final α_t the lowerbound proven above. We check below that this smaller value of α_t still guarantees a decrease in the function value that is sufficient for the statement of the theorem to hold.

A consistency check! Let us substitute the above final value of the step length $\alpha_t = \frac{1}{L} \frac{\|\mathbf{g}_t\|^2 \left(\frac{\sigma^2(\beta_1 - \beta_1^2)}{\xi\sigma(\xi+\sigma)}\right)}{\left(\frac{(1-\beta_1^t)\sigma}{\xi}\right)^2} = \frac{\xi}{L(1-\beta_1^t)^2} \|\mathbf{g}_t\|^2 \left(\frac{(\beta_1 - \beta_1^2)}{\sigma(\xi+\sigma)}\right)$, the bound in equation 5.16 (with σ_t replaced by σ), and the bound in equation 5.20 (at the chosen values of $\theta_1 = 1$ and $\theta_2 = \sigma$) in the original equation 5.14 to measure the decrease in the function value between consecutive steps,

$$f(\mathbf{x}_{t+1}) - f(\mathbf{x}_t)$$
$$\leq \alpha_t \left(-\langle \mathbf{g}_t, \left(V_t^{\frac{1}{2}} + \text{diag}(\xi \mathbf{1}_d)\right)^{-1} \mathbf{m}_t \rangle + \frac{L\alpha_t}{2} \| \left(V_t^{\frac{1}{2}} + \text{diag}(\xi \mathbf{1}_d)\right)^{-1} \mathbf{m}_t \|^2 \right)$$
$$\leq \alpha_t \left(-Q_t + \frac{L\alpha_t}{2} \| \left(V_t^{\frac{1}{2}} + \text{diag}(\xi \mathbf{1}_d)\right)^{-1} \mathbf{m}_t \|^2 \right)$$
$$\leq \frac{\xi}{L(1-\beta_1^t)^2} \|\mathbf{g}_t\|^2 \left(\frac{(\beta_1 - \beta_1^2)}{\sigma(\xi+\sigma)}\right) \left(-\|\mathbf{g}_t\|^2 \left(\frac{\sigma(\beta_1 - \beta_1^2)\theta_1\theta_2}{\xi\sigma(\xi+\sigma)}\right)\right)$$
$$+ \frac{L}{2} \left(\frac{\xi}{L(1-\beta_1^t)^2} \|\mathbf{g}_t\|^2 \left(\frac{(\beta_1 - \beta_1^2)}{\sigma(\xi+\sigma)}\right) \frac{(1-\beta_1^t)\sigma}{\xi} \right)^2$$

The RHS above can be simplified to be shown to be equal to the RHS in equation 5.21 at the same values of θ_1 and θ_2 as used above. And we recall that the bound on the running time was derived from this equation 5.21. □

5.8 Conclusion

To the best of our knowledge, we present the first theoretical guarantees of convergence to criticality for the immensely popular algorithms RMSProp and ADAM in their most commonly used setting of optimizing a non-convex objective.

By our experiments, we have sought to shed light on the important topic of the interplay between adaptivity and momentum in training nets. By choosing to study textbook autoencoder architectures where various parameters of the net can be changed controllably we highlight the following two aspects that (a) the value of the gradient shifting hyperparameter ξ has a significant influence on the performance of ADAM and RMSProp and (b) ADAM seems to perform particularly well (often supersedes Nesterov accelerated gradient method) when its momentum parameter β_1 is very close to 1. On VGG-9 with CIFAR-10 and for the task of training autoencoders on MNIST we have verified these conclusions across different widths and depths of nets as well as in the full-batch and the mini-batch setting (with large nets) and also under compression of the input/output image size.

Curiously enough, this regime of β_1 being close to 1 is currently not within the reach of our proof techniques of showing convergence for ADAM. Our experiments give strong reasons to try to advance theory in this direction in future work. Though we note that it is still open to find a characterization of the class of objectives for which ADAM and RMSProp in their standard stochastic forms converge to criticality using just a bounded moment and unbiased gradient estimating oracle. Hence theoretically we are still far from being able to explain the unique advantages of the standard versions of RMSProp or ADAM, which in turn we have thoroughly demonstrated in the experiments in this work.

Appendix To Chapter 5

5.A Proving stochastic RMSProp (Proof of Theorem 5.3.3)

Proof. We define $\sigma_t := \max_{k=1,..,t} \|\nabla f_{i_k}(\mathbf{x}_k)\|$ and we solve the recursion for $\mathbf{v}_t$ as, $\mathbf{v}_t = (1-\beta_2)\sum_{k=1}^{t} \beta_2^{t-k}(\mathbf{g}_k^2 + \xi)$. This lets us write the following bounds,

$$\lambda_{min}(V_t^{-\frac{1}{2}}) \geq \frac{1}{\sqrt{\max_{i=1,..,d}(\mathbf{v}_t)_i}} \geq \frac{1}{\sqrt{\max_{i=1,..,d}((1-\beta_2)\sum_{k=1}^{t}\beta_2^{t-k}(\mathbf{g}_k^2 + \xi \mathbb{1}_d)_i)}}$$
$$\geq \frac{1}{\sqrt{1-\beta_2^t}\sqrt{\sigma_t^2 + \xi}}$$

Now we define, $\epsilon_t := \min_{k=1,..,t,i=1,..,d}(\nabla f_{i_k}(\mathbf{x}_k))_i^2$ and this lets us get the following bounds,

$$\lambda_{max}(V_t^{-\frac{1}{2}}) \leq \frac{1}{\min_{i=1,..,d}(\sqrt{(\mathbf{v}_t)_i})} \leq \frac{1}{\sqrt{(1-\beta_2^t)}\sqrt{(\xi + \epsilon_t)}}$$

Now we invoke the bounded gradient assumption about the f_i functions and replace in the above equation the eigenvalue bounds of the pre-conditioner by worst-case estimates $\mu_{\max}$ and $\mu_{\min}$ defined as,

$$\lambda_{min}(V_t^{-\frac{1}{2}}) \geq \frac{1}{\sqrt{\sigma_f^2 + \xi}} := \mu_{\min}$$
$$\lambda_{max}(V_t^{-\frac{1}{2}}) \leq \frac{1}{\sqrt{(1-\beta_2)}\sqrt{\xi}} := \mu_{max}$$

Using the L-smoothness of f between consecutive iterates $\mathbf{x}_t$ and $\mathbf{x}_{t+1}$ we have,

$$f(\mathbf{x}_{t+1}) \leq f(\mathbf{x}_t) + \langle \nabla f(\mathbf{x}_t), \mathbf{x}_{t+1} - \mathbf{x}_t \rangle + \frac{L}{2}\|\mathbf{x}_{t+1} - \mathbf{x}_t\|^2$$

We note that the update step of stochastic RMSProp is $x_{t+1} = x_t - \alpha(V_t)^{-\frac{1}{2}} g_t$ where g_t is the stochastic gradient at iterate x_t. Let $H_t = \{\mathbf{x}_1, \mathbf{x}_2, .., \mathbf{x}_t\}$ be the set of random variables corresponding to the first t iterates. The assumptions we have about the stochastic oracle give us the following relations, $\mathbb{E}[g_t \mid H_t] = \nabla f(x_t)$ and $\mathbb{E}[\|g_t\|^2 \mid H_t] \leq \sigma_f^2$. Now we can invoke these stochastic oracle's properties and take a conditional (on H_t) expectation over $\mathbf{g}_t$ of the $L-$smoothness in equation to get,

$$\begin{aligned}
\mathbb{E}[f(\mathbf{x}_{t+1}) \mid H_t] &\leq f(\mathbf{x}_t) - \alpha \mathbb{E}\left[\langle \nabla f(\mathbf{x}_t), (V_t)^{-\frac{1}{2}} \mathbf{g}_t \rangle \mid H_t\right] + \frac{\alpha^2 L}{2} \mathbb{E}\left[\|(V_t)^{-\frac{1}{2}} \mathbf{g}_t\|^2 \mid H_t\right] \\
&\leq f(\mathbf{x}_t) - \alpha \mathbb{E}\left[\langle \nabla f(\mathbf{x}_t), (V_t)^{-\frac{1}{2}} \mathbf{g}_t \rangle \mid H_t\right] + \mu_{\max}^2 \frac{\alpha^2 L}{2} \mathbb{E}\left[\|\mathbf{g}_t\|^2 \mid H_t\right] \\
&\leq f(\mathbf{x}_t) - \alpha \mathbb{E}\left[\langle \nabla f(\mathbf{x}_t), (V_t)^{-\frac{1}{2}} \mathbf{g}_t \rangle \mid H_t\right] + \mu_{\max}^2 \frac{\alpha^2 \sigma_f^2 L}{2}
\end{aligned} \tag{5.22}$$

We now separately analyze the middle term in the RHS above. In Lemma 5.A.1 below and we get,

$$\mathbb{E}[\langle \nabla f(\mathbf{x}_t), (V_t)^{-\frac{1}{2}} \mathbf{g}_t \rangle \mid H_t] \geq \mu_{min} \|\nabla f(\mathbf{x}_t)\|^2$$

We substitute the above into equation 5.22 and take expectations over H_t to get,

$$\begin{aligned}
\mathbb{E}[f(\mathbf{x}_{t+1}) - f(\mathbf{x}_t)] &\leq -\alpha \mu_{min} \mathbb{E}[\|\nabla f(\mathbf{x}_t)\|^2] + \mu_{max}^2 \frac{\alpha^2 \sigma_f^2 L}{2} \\
\implies \mathbb{E}[\|\nabla f(\mathbf{x}_t)\|^2] &\leq \frac{1}{\alpha \mu_{min}} \mathbb{E}[f(\mathbf{x}_t) - f(\mathbf{x}_{t+1})] + \frac{\alpha \sigma_f^2 L}{2} \frac{\mu_{max}^2}{\mu_{min}}
\end{aligned} \tag{5.23}$$

Doing the above replacements to upperbound the RHS of equation 5.23 and summing the inequation over $t = 1$ to $t = T$ and taking the average and replacing the LHS by a lowerbound of it, we get,

$$\begin{aligned}\min_{t=1\ldots T} \mathbb{E}[\|\nabla f(\mathbf{x}_t)\|^2] &\leq \frac{1}{\alpha T \mu_{\min}} \mathbb{E}[f(\mathbf{x}_1) - f(\mathbf{x}_{T+1})] + \frac{\alpha \sigma_f^2 L}{2} \frac{\mu_{\max}^2}{\mu_{\min}} \\ &\leq \frac{1}{\alpha T \mu_{\min}} (f(\mathbf{x}_1) - f(\mathbf{x}_*)) + \frac{\alpha \sigma_f^2 L}{2} \frac{\mu_{\max}^2}{\mu_{\min}}\end{aligned}$$

Replacing into the RHS above the optimal choice of,

$$\alpha = \frac{1}{\sqrt{T}} \sqrt{\frac{2\,(f(\mathbf{x}_1) - f(\mathbf{x}_*))}{\sigma_f^2 L \mu_{\max}^2}} = \frac{1}{\sqrt{T}} \sqrt{\frac{2\xi(1-\beta_2)\,(f(\mathbf{x}_1) - f(\mathbf{x}_*))}{\sigma_f^2 L}}$$

we get,

$$\begin{aligned}\min_{t=1\ldots T} \mathbb{E}[\|\nabla f(\mathbf{x}_t)\|^2] &\leq 2\sqrt{\frac{1}{T\mu_{\min}} (f(\mathbf{x}_1) - f(\mathbf{x}_*)) \times \frac{L\sigma_f^2}{2} \frac{\mu_{\max}^2}{\mu_{\min}}} \\ &= \frac{1}{\sqrt{T}} \sqrt{\frac{2L\sigma_f^2(\sigma_f^2 + \xi)\,(f(\mathbf{x}_1) - f(\mathbf{x}_*))}{(1-\beta_2)\xi}}\end{aligned}$$

Thus stochastic RMSProp with the above step-length is guaranteed is reach ϵ criticality in number of iterations given by, $T \leq \frac{1}{\epsilon^4} \left(\frac{2L\sigma_f^2(\sigma_f^2+\xi)(f(\mathbf{x}_1)-f(\mathbf{x}_*))}{(1-\beta_2)\xi} \right)$ □

Lemma 5.A.1. At any time t, the following holds,

$$\mathbb{E}[\langle \nabla f(x_t), V_t^{-1/2} g_t \rangle \mid H_t] \geq \mu_{\min} \|\nabla f(x_t)\|^2$$

Proof.

$$\begin{aligned}\mathbb{E}\left[\langle \nabla f(x_t), V_t^{-1/2} g_t \rangle \mid H_t\right] &= \mathbb{E}\left[\sum_{i=1}^{d} \nabla_i f(x_t) (V_t^{-1/2})_{ii} (g_t)_i \mid H_t\right] \\ &= \sum_{i=1}^{d} \nabla_i f(x_t) \mathbb{E}\left[(V_t^{-1/2})_{ii} (g_t)_i \mid H_t\right] \qquad (5.24)\end{aligned}$$

Now we introduce some new variables to make the analysis easier to present. Let $a_{pi} := \left[\nabla f_p(\mathbf{x}_t)\right]_i$ where p indexes the training data set, $p \in \{1, \ldots, k\}$. (conditioned on H_t, a_{pi}s are constants) This implies, $\nabla_i f(x_t) = \frac{1}{k} \sum_{p=1}^{k} a_{pi}$ We recall that $\mathbb{E}\left[(\mathbf{g}_t)_i\right] = \nabla_i f(\mathbf{x}_t)$ where the expectation is taken over the oracle call at the t^{th} update step. Further our instantiation of the oracle is equivalent to doing the uniformly at random sampling, $(\mathbf{g}_t)_i \sim \{a_{pi}\}_{p=1,\ldots,k}$.

Given that we have, $V_t = \text{diag}(\mathbf{v}_t)$ with $\mathbf{v}_t = (1-\beta_2) \sum_{k=1}^{t} \beta_2^{t-k} (\mathbf{g}_k^2 + \xi \mathbf{1}_d)$ this implies, $(V_t^{-1/2})_{ii} =$

$\frac{1}{\sqrt{(1-\beta_2)(\mathbf{g}_i)_i^2+d_i}}$ where we have defined $d_i := (1-\beta_2)\xi + (1-\beta_2)\sum_{k=1}^{t-1}\beta_2^{t-k}((\mathbf{g}_k)_i^2+\xi)$. (conditioned on H_t, d_i is a constant) This leads to an explicit form of the needed expectation over the $t^{th}-$oracle call as,

$$\begin{aligned}\mathbb{E}\left[(V_t^{-1/2})_{ii}(g_t)_i \mid H_t\right] &= \mathbb{E}\left[(V_t^{-1/2})_{ii}(g_t)_i \mid H_t\right] \\ &= \mathbb{E}_{(\mathbf{g}_t)_i\sim\{a_{pi}\}_{p=1,\dots,k}}\left[\frac{(g_t)_i}{\sqrt{(1-\beta_2)(g_t)_i^2+d_i}} \mid H_t\right] \\ &= \frac{1}{k}\sum_{p=1}^{k}\frac{a_{pi}}{\sqrt{(1-\beta_2)a_{pi}^2+d_i}}\end{aligned}$$

Substituting the above (and the definition of the constants a_{pi}) back into equation 5.24 we have,

$$\mathbb{E}\left[\langle\nabla f(x_t), V_t^{-1/2}g_t\rangle \mid H_t\right] = \sum_{i=1}^{d}\left(\frac{1}{k}\sum_{p=1}^{k}a_{pi}\right)\left(\frac{1}{k}\sum_{p=1}^{k}\frac{a_{pi}}{\sqrt{(1-\beta_2)a_{pi}^2+d_i}}\right)$$

We define two vectors $\mathbf{a}_i, \mathbf{h}_i \in \mathbb{R}^k$ s.t $(\mathbf{a}_i)_p = a_{pi}$ and $(\mathbf{h}_i)_p = \frac{1}{\sqrt{(1-\beta_2)a_{pi}^2+d_i}}$

Substituting this, the above expression can be written as,

$$\mathbb{E}\left[\langle\nabla f(x_t), V_t^{-1/2}g_t\rangle \mid H_t\right] = \frac{1}{k^2}\sum_{i=1}^{d}\left(\mathbf{a}_i^\top\mathbf{1}_k\right)\left(\mathbf{h}_i^\top\mathbf{a}_i\right) = \frac{1}{k^2}\sum_{i=1}^{d}\mathbf{a}_i^\top\left(\mathbf{1}_k\mathbf{h}_i^\top\right)\mathbf{a}_i \tag{5.25}$$

Note that with this substitution, the RHS of the claimed lemma becomes,

$$\begin{aligned}\mu_{\min}\|\nabla f(x_t)\|^2 &= \mu_{\min}\sum_{i=1}^{d}\left(\frac{1}{k}\sum_{p=1}^{k}\nabla_p f(\mathbf{x}_t)\right)^2 \\ &= \frac{\mu_{\min}}{k^2}\sum_{i=1}^{d}(\mathbf{a}_i^\top\mathbf{1}_k)^2 \\ &= \frac{\mu_{\min}}{k^2}\sum_{i=1}^{d}\mathbf{a}_i^\top\mathbf{1}_k\mathbf{1}_k^\top\mathbf{a}_i\end{aligned}$$

Therefore our claim is proved if we show that for all i,

$$\frac{1}{k^2}\mathbf{a}_i^\top\left(\mathbf{1}_k\mathbf{h}_i^\top\right)\mathbf{a}_i - \frac{\mu_{\min}}{k^2}\mathbf{a}_i^\top\mathbf{1}_k\mathbf{1}_k^\top\mathbf{a}_i \geq 0$$

. This can be simplified as,

$$\frac{1}{k^2}\mathbf{a}_i^\top\left(\mathbf{1}_k\mathbf{h}_i^\top\right)\mathbf{a}_i - \mu_{\min}\frac{1}{k^2}\mathbf{a}_i^\top\mathbf{1}_k\mathbf{1}_k^\top\mathbf{a}_i = \frac{1}{k^2}\mathbf{a}_i^\top\left(\mathbf{1}_k\left(\mathbf{h}_i - \mu_{\min}\mathbf{1}_k\right)^\top\right)\mathbf{a}_i$$

To further simplify, we define $\mathbf{q}_i \in \mathbb{R}^k, (\mathbf{q}_i)_p = (\mathbf{h}_i)_p - \mu_{\min} = \frac{1}{\sqrt{(1-\beta_2)a_{pi}^2+d_i}} - \mu_{\min}$. We therefore need to show,

$$\frac{1}{k^2}\mathbf{a}_i^\top\left(\mathbf{1}_k\mathbf{q}_i^\top\right)\mathbf{a}_i \geq 0$$

We first bound d_i by recalling the definition of σ_f (from which it follows that $a_{pi}^2 \leq \sigma_f^2$),

$$\begin{aligned}
&d_i \leq (1-\beta_2)\left[\xi + \sum_{k=1}^{t-1}\beta_2^{t-k}(\sigma_f^2+\xi)\right] = (1-\beta_2)\left[\xi + \frac{\beta_2-\beta_2^{t-1}}{1-\beta_2}(\sigma_f^2+\xi)\right] \\
&\leq (1-\beta_2)\xi + (\beta_2-\beta_2^{t-1})\xi + (\beta_2-\beta_2^{t-1})\sigma_f^2 = (1-\beta_2^{t-1})\xi + (\beta_2-\beta_2^{t-1})\sigma_f^2 \\
&\implies \sqrt{(1-\beta_2)a_{pi}^2+d_i} \leq \sqrt{(1-\beta_2)\sigma_f^2 + (1-\beta_2^{t-1})\xi + (\beta_2-\beta_2^{t-1})\sigma_f^2} \\
&= \sqrt{(1-\beta_2^{t-1})(\sigma_f^2+\xi)} \\
&\implies -\mu_{\min} + \frac{1}{\sqrt{(1-\beta_2)a_{pi}^2+d_i}} \geq -\mu_{\min} + \frac{1}{\sqrt{(1-\beta_2^{t-1})(\sigma_f^2+\xi)}} \\
&= -\frac{1}{\sqrt{\sigma_f^2+\xi}} + \frac{1}{\sqrt{(1-\beta_2^{t-1})(\sigma_f^2+\xi)}} \\
&\implies -\mu_{\min} + \frac{1}{\sqrt{(1-\beta_2)a_{pi}^2+d_i}} \geq 0
\end{aligned} \tag{5.26}$$

The inequality follows since $\beta_2 \in (0, 1]$

Putting this all together, we get,

$$\begin{aligned}
&(\mathbf{a}_i^\top \mathbf{1}_k)(\mathbf{q}_i^\top \mathbf{a}_i) \\
&= \left(\sum_{p=1}^{k} a_{pi}\right)\left(\sum_{p=1}^{k}\left[-\mu_{\min} a_{pi} + \frac{a_{pi}}{\sqrt{(1-\beta_2)a_{pi}^2 + d_i}}\right]\right) \\
&= \sum_{p,q=1}^{k}\left[-\mu_{\min} a_{pi}a_{qi} + \frac{a_{pi}a_{qi}}{\sqrt{(1-\beta_2)a_{pi}^2 + d_i}}\right] \\
&= \sum_{p,q=1}^{k} a_{pi}a_{qi}\left[-\mu_{\min} + \frac{1}{\sqrt{(1-\beta_2)a_{pi}^2 + d_i}}\right]
\end{aligned}$$

Now our assumption that for all $\mathbf{x}$, $\text{sign}(\nabla f_p(\mathbf{x})) = \text{sign}(\nabla f_q(\mathbf{x}))$ for all $p, q \in \{1, \ldots, k\}$ leads to the conclusion that the term $a_{pi}a_{qi} \geq 0$. And we had already shown in equation 5.26 that $\left[-\mu_{\min} + \frac{1}{\sqrt{(1-\beta_2)a_{pi}^2 + d_i}}\right] \geq 0$. Thus we have shown that $(\mathbf{a}_i^\top \mathbf{1}_k)(\mathbf{q}_i^\top \mathbf{a}_i) \geq 0$ and this finishes the proof. □

5.B Proving deterministic RMSProp - the version with standard speed (Proof of Theorem 5.4.1)

Proof. By the $L-$smoothness condition and the update rule in Algorithm 6 we have,

$$\begin{aligned}
&f(\mathbf{x}_{t+1}) \leq f(\mathbf{x}_t) - \alpha_t \langle \nabla f(\mathbf{x}_t), V_t^{-\frac{1}{2}} \nabla f(\mathbf{x}_t)\rangle + \alpha_t^2 \frac{L}{2} \|V_t^{-\frac{1}{2}} \nabla f(\mathbf{x}_t)\|^2 \\
\implies &f(\mathbf{x}_{t+1}) - f(\mathbf{x}_t) \leq \alpha_t \left(\frac{L\alpha_t}{2}\|V_t^{-\frac{1}{2}} \nabla f(\mathbf{x}_t)\|^2 - \langle \nabla f(\mathbf{x}_t), V_t^{-\frac{1}{2}} \nabla f(\mathbf{x}_t)\rangle\right) \qquad (5.27)
\end{aligned}$$

For $0 < \delta_t^2 < \frac{1}{\sqrt{1-\beta_2^t}\sqrt{\sigma_t^2 + \xi}}$ we now show a strictly positive lowerbound on the following function,

$$\frac{2}{L}\left(\frac{\langle \nabla f(\mathbf{x}_t), V_t^{-\frac{1}{2}} \nabla f(\mathbf{x}_t)\rangle - \delta_t^2 \|\nabla f(\mathbf{x}_t)\|^2}{\|V_t^{-\frac{1}{2}} \nabla f(\mathbf{x}_t)\|^2}\right) \qquad (5.28)$$

We define $\sigma_t := \max_{i=1,\ldots,t} \|\nabla f(\mathbf{x}_i)\|$ and we solve the recursion for $\mathbf{v}_t$ as, $\mathbf{v}_t = (1-\beta_2)\sum_{k=1}^{t} \beta_2^{t-k}(\mathbf{g}_k^2 + \xi)$. This lets us write the following bounds,

$$\begin{aligned}\langle \nabla f(\mathbf{x}_t), V_t^{-\frac{1}{2}} \nabla f(\mathbf{x}_t) \rangle &\geq \lambda_{min}(V_t^{-\frac{1}{2}}) \|\nabla f(\mathbf{x}_t)\|^2 \geq \frac{\|\nabla f(\mathbf{x}_t)\|^2}{\sqrt{\max_{i=1,..,d}(\mathbf{v}_t)_i}} \\ &\geq \frac{\|\nabla f(\mathbf{x}_t)\|^2}{\sqrt{\max_{i=1,..,d}((1-\beta_2)\sum_{k=1}^{t} \beta_2^{t-k}(\mathbf{g}_k^2 + \xi \mathbb{1}_d)_i)}} \\ &\geq \frac{\|\nabla f(\mathbf{x}_t)\|^2}{\sqrt{1-\beta_2^t}\sqrt{\sigma_t^2 + \xi}} \end{aligned} \tag{5.29}$$

Now we define, $\epsilon_t := \min_{k=1,..,t,i=1,..,d}(\nabla f(\mathbf{x}_k))_i^2$ and this lets us get the following sequence of inequalities,

$$\|V_t^{-\frac{1}{2}} \nabla f(\mathbf{x}_t)\|^2 \leq \lambda_{max}^2(V_t^{-\frac{1}{2}}) \|\nabla f(\mathbf{x}_t)\|^2 \leq \frac{\|\nabla f(\mathbf{x}_t)\|^2}{(\min_{i=1,..,d}(\sqrt{(\mathbf{v}_t)_i}))^2} \tag{5.30}$$

$$\leq \frac{\|\nabla f(\mathbf{x}_t)\|^2}{(1-\beta_2^t)(\xi + \epsilon_t)} \tag{5.31}$$

So combining equations 5.30 and 5.29 into equation 5.28 and from the exit line in the loop we are assured that $\|\nabla f(\mathbf{x}_t)\|^2 \neq 0$ and combining these we have,

$$\begin{aligned}&\frac{2}{L}\left(\frac{-\delta_t^2 \|\nabla f(\mathbf{x}_t)\|^2 + \langle \nabla f(\mathbf{x}_t), V_t^{-\frac{1}{2}} \nabla f(\mathbf{x}_t) \rangle}{\|V_t^{-\frac{1}{2}} \nabla f(\mathbf{x}_t)\|^2}\right) \\ &\geq \frac{2}{L}\left(\frac{-\delta_t^2 + \frac{1}{\sqrt{1-\beta_2^t}\sqrt{\sigma_t^2+\xi}}}{\frac{1}{(1-\beta_2^t)(\xi+\epsilon_t)}}\right) \\ &\geq \frac{2(1-\beta_2^t)(\xi+\epsilon_t)}{L}\left(-\delta_t^2 + \frac{1}{\sqrt{1-\beta_2^t}\sqrt{\sigma_t^2+\xi}}\right)\end{aligned}$$

Now our definition of δ_t^2 allows us to define a parameter $0 < \beta_t := \frac{1}{\sqrt{1-\beta_2^t}\sqrt{\sigma_t^2+\xi}} - \delta_t^2$ and rewrite the above equation as,

$$\frac{2}{L}\left(\frac{-\delta_t^2 \|\nabla f(\mathbf{x}_t)\|^2 + \langle \nabla f(\mathbf{x}_t), V_t^{-\frac{1}{2}} \nabla f(\mathbf{x}_t) \rangle}{\|V_t^{-\frac{1}{2}} \nabla f(\mathbf{x}_t)\|^2}\right) \geq \frac{2(1-\beta_2^t)(\xi+\epsilon_t)\beta_t}{L} \tag{5.32}$$

We can as well satisfy the conditions needed on the variables, β_t and δ_t by choosing,

$$\delta_t^2 = \frac{1}{2}\min_{t=1,\dots}\frac{1}{\sqrt{1-\beta_2^t}\sqrt{\sigma_t^2+\xi}} = \frac{1}{2}\frac{1}{\sqrt{\sigma^2+\xi}} =: \delta^2$$

and

$$\beta_t = \min_{t=1,..}\frac{1}{\sqrt{1-\beta_2^t}\sqrt{\sigma_t^2+\xi}} - \delta^2 = \frac{1}{2}\frac{1}{\sqrt{\sigma^2+\xi}}$$

Then the worst-case lowerbound in equation 5.32 becomes,

$$\frac{2}{L}\left(\frac{-\delta_t^2\|\nabla f(\mathbf{x}_t)\|^2 + \langle \nabla f(\mathbf{x}_t), V_t^{-\frac{1}{2}}\nabla f(\mathbf{x}_t)\rangle}{\|V_t^{-\frac{1}{2}}\nabla f(\mathbf{x}_t)\|^2}\right) \geq \frac{2(1-\beta_2)\xi}{L} \times \frac{1}{2}\frac{1}{\sqrt{\sigma^2+\xi}}$$

This now allows us to see that a constant step length $\alpha_t = \alpha > 0$ can be defined as, $\alpha = \frac{(1-\beta_2)\xi}{L\sqrt{\sigma^2+\xi}}$ and this is such that the above equation can be written as, $\frac{L\alpha}{2}\|V_t^{-\frac{1}{2}}\nabla f(\mathbf{x}_t)\|^2 - \langle \nabla f(\mathbf{x}_t), V_t^{-\frac{1}{2}}\nabla f(\mathbf{x}_t)\rangle \leq -\delta^2\|\nabla f(\mathbf{x}_t)\|^2$. This when substituted back into equation 5.27 we have,

$$f(\mathbf{x}_{t+1}) - f(\mathbf{x}_t) \leq -\delta^2\alpha\|\nabla f(\mathbf{x}_t)\|^2 = -\delta^2\alpha\|\nabla f(\mathbf{x}_t)\|^2$$

This gives us,

$$\begin{aligned}
\|\nabla f(\mathbf{x}_t)\|^2 &\leq \frac{1}{\delta^2\alpha}[f(\mathbf{x}_t) - f(\mathbf{x}_{t+1})] \\
\implies \sum_{t=1}^{T}\|\nabla f(\mathbf{x}_t)\|^2 &\leq \frac{1}{\delta^2\alpha}[f(\mathbf{x}_1) - f(\mathbf{x}_*)] \qquad (5.33)\\
\implies \min_{t=1,,.T}\|\nabla f(\mathbf{x}_t)\|^2 &\leq \frac{1}{T\delta^2\alpha}[f(\mathbf{x}_1) - f(\mathbf{x}_*)] \qquad (5.34)
\end{aligned}$$

Thus for any given $\epsilon > 0$, T satisfying, $\frac{1}{T\delta^2\alpha}[f(\mathbf{x}_1) - f(\mathbf{x}_*)] \leq \epsilon^2$ is a sufficient condition to ensure that the algorithm finds a point $\mathbf{x}_{result} := \text{argmin}_{t=1,,.T}\|\nabla f(\mathbf{x}_t)\|^2$ with $\|\nabla f(\mathbf{x}_{result})\|^2 \leq \epsilon^2$.

Thus we have shown that using a constant step length of $\alpha = \frac{(1-\beta_2)\xi}{L\sqrt{\sigma^2+\xi}}$ deterministic RMSProp can find an $\epsilon-$critical point in $T = \frac{1}{\epsilon^2} \times \frac{f(\mathbf{x}_1)-f(\mathbf{x}_*)}{\delta^2\alpha} = \frac{1}{\epsilon^2} \times \frac{2L(\sigma^2+\xi)(f(\mathbf{x}_1)-f(\mathbf{x}_*))}{(1-\beta_2)\xi}$ steps.

□

5.C Proving deterministic RMSProp - the version with no added shift (Proof of Theorem 5.4.2)

Proof. From the $L-$smoothness condition on f we have between consecutive iterates of the above algorithm,

$$f(\mathbf{x}_{t+1}) \leq f(\mathbf{x}_t) - \alpha_t \langle \nabla f(\mathbf{x}_t), V_t^{-\frac{1}{2}} \nabla f(\mathbf{x}_t) \rangle + \frac{L}{2} \alpha_t^2 \| V_t^{-\frac{1}{2}} \nabla f(\mathbf{x}_t) \|^2 \tag{5.35}$$

$$\implies \langle \nabla f(\mathbf{x}_t), V_t^{-\frac{1}{2}} \nabla f(\mathbf{x}_t) \rangle \leq \frac{1}{\alpha_t} \left(f(\mathbf{x}_t) - f(\mathbf{x}_{t+1}) \right) + \frac{L \alpha_t}{2} \| V_t^{-\frac{1}{2}} \nabla f(\mathbf{x}_t) \|^2 \tag{5.36}$$

Now the recursion for $\mathbf{v}_t$ can be solved to get, $\mathbf{v}_t = (1-\beta_2) \sum_{k=1}^{t} \beta_2^{t-k} \mathbf{g}_k^2$. Then

$$\begin{aligned} \| V_t^{\frac{1}{2}} \| &\geq \frac{1}{\max_{i \in \text{Support}(\mathbf{v}_t)} \sqrt{(\mathbf{v}_t)_i}} \\ &= \frac{1}{\max_{i \in \text{Support}(\mathbf{v}_t)} \sqrt{(1-\beta_2) \sum_{k=1}^{t} \beta_2^{t-k} (\mathbf{g}_k^2)_i}} \\ &= \frac{1}{\max_{i \in \text{Support}(\mathbf{v}_t)} \sigma \sqrt{(1-\beta_2) \sum_{k=1}^{t} \beta_2^{t-k}}} = \frac{1}{\sigma \sqrt{(1-\beta_2^t)}} \end{aligned}$$

Substituting this in a lowerbound on the LHS of equation 5.35 we get,

$$\frac{1}{\sigma \sqrt{(1-\beta_2^t)}} \| \nabla f(\mathbf{x}_t) \|^2 \leq \langle \nabla f(\mathbf{x}_t), V_t^{-\frac{1}{2}} \nabla f(\mathbf{x}_t) \rangle \leq \frac{1}{\alpha_t} \left(f(\mathbf{x}_t) - f(\mathbf{x}_{t+1}) \right) + \frac{L \alpha_t}{2} \| V_t^{-\frac{1}{2}} \nabla f(\mathbf{x}_t) \|^2$$

Summing the above we get,

$$\sum_{t=1}^{T} \frac{1}{\sigma \sqrt{(1-\beta_2^t)}} \| \nabla f(\mathbf{x}_t) \|^2 \leq \sum_{t=1}^{T} \frac{1}{\alpha_t} \left(f(\mathbf{x}_t) - f(\mathbf{x}_{t+1}) \right) + \sum_{t=1}^{T} \frac{L \alpha_t}{2} \| V_t^{-\frac{1}{2}} \nabla f(\mathbf{x}_t) \|^2 \tag{5.37}$$

Now we substitute $\alpha_t = \frac{\alpha}{\sqrt{t}}$ and invoke the definition of B_ℓ and B_u to write the first term on the RHS of equation 5.37 as,

$$\begin{aligned}\sum_{t=1}^{T}\frac{1}{\alpha_t}[f(\mathbf{x}_t)-f(\mathbf{x}_{t+1})] &= \frac{f(\mathbf{x}_1)}{\alpha}+\sum_{t=1}^{T}\left(\frac{f(\mathbf{x}_{t+1})}{\alpha_{t+1}}-\frac{f(\mathbf{x}_{t+1})}{\alpha_t}\right)-\frac{f(x_{T+1})}{\alpha_{T+1}}\\ &= \frac{f(\mathbf{x}_1)}{\alpha}-\frac{f(x_{T+1})}{\alpha_{T+1}}+\frac{1}{\alpha}\sum_{t=1}^{T}f(\mathbf{x}_{t+1})(\sqrt{t+1}-\sqrt{t})\\ &\leq \frac{B_u}{\alpha}-\frac{B_\ell\sqrt{T+1}}{\alpha}+\frac{B_u}{\alpha}(\sqrt{T+1}-1)\end{aligned}$$

Now we bound the second term in the RHS of equation 5.37 as follows. Lets first define a function $P(T)$ as follows, $P(T)=\sum_{t=1}^{T}\alpha_t\|V_t^{-\frac{1}{2}}\nabla f(\mathbf{x}_t)\|^2$ and that gives us,

$$\begin{aligned}P(T)-P(T-1) &= \alpha_T\sum_{i=1}^{d}\frac{\mathbf{g}_{T,i}^2}{\mathbf{v}_{T,i}}=\alpha_T\sum_{i=1}^{d}\frac{\mathbf{g}_{T,i}^2}{(1-\beta_2)\sum_{k=1}^{T}\beta_2^{T-k}\mathbf{g}_{k,i}^2}\\ &= \frac{\alpha_T}{(1-\beta_2)}\sum_{i=1}^{d}\frac{\mathbf{g}_{T,i}^2}{\sum_{k=1}^{T}\beta_2^{T-k}\mathbf{g}_{k,i}^2}\leq\frac{d\alpha}{(1-\beta_2)\sqrt{T}}\\ \implies \sum_{t=2}^{T}[P(t)-P(t-1)] &= P(T)-P(1)\\ &\leq \frac{d\alpha}{(1-\beta_2)}\sum_{t=2}^{T}\frac{1}{\sqrt{t}}\leq\frac{d\alpha}{2(1-\beta_2)}(\sqrt{T}-2)\\ \implies P(T) &\leq P(1)+\frac{d\alpha}{2(1-\beta_2)}(\sqrt{T}-2)\end{aligned}$$

So substituting the above two bounds back into the RHS of the above inequality 5.37and removing the factor of $\sqrt{1-\beta_2^T}<1$ from the numerator, we can define a point $\mathbf{x}_{result}$ as follows,

$$\begin{aligned}\|\nabla f(\mathbf{x}_{result})\|^2 := \underset{t=1,..,T}{\operatorname{argmin}}\|\nabla f(\mathbf{x}_t)\|^2 &\leq \frac{1}{T}\sum_{t=1}^{T}\|\nabla f(\mathbf{x}_t)\|^2\\ &\leq \frac{\sigma}{T}\left(\frac{B_u}{\alpha}-\frac{B_l\sqrt{T+1}}{\alpha}+\frac{B_u}{\alpha}(\sqrt{T+1}-1)\right.\\ &\quad\left.+\frac{L}{2}\left[P(1)+\frac{d\alpha}{2(1-\beta_2)}(\sqrt{T}-2)\right]\right)\end{aligned}$$

Thus it follows that for $T=O(\frac{1}{\epsilon^4})$ the algorithm 6 is guaranteed to have found at least one point $\mathbf{x}_{result}$ such that, $\|\nabla f(\mathbf{x}_{result})\|^2\leq\epsilon^2$ □

5.D Hyperparameter Tuning

Here we describe how we tune the hyper-parameters of each optimization algorithm. NAG has two hyper-parameters, the step size α and the momentum μ. The main hyper-parameters for RMSProp are the step size α, the decay parameter β_2 and the perturbation ξ. ADAM, in addition to the ones in RMSProp, also has a momentum parameter β_1. We vary the step-sizes of ADAM in the conventional way of $\alpha_t = \alpha\sqrt{1-\beta_2^t}/(1-\beta_1^t)$.

For tuning the step size, we follow the same method used in Wilson et al., 2017. We start out with a logarithmically-spaced grid of five step sizes. If the best performing parameter was at one of the extremes of the grid, we tried new grid points so that the best performing parameters were at one of the middle points in the grid. While it is computationally infeasible even with substantial resources to follow a similarly rigorous tuning process for all other hyper-parameters, we do tune over them somewhat as described below.

NAG The initial set of step sizes used for NAG were: $\{3e{-}3, 1e{-}3, 3e{-}4, 1e{-}4, 3e{-}5\}$ We tune the momentum parameter over values $\mu \in \{0.9, 0.99\}$.

RMSProp The initial set of step sizes used were: $\{3e{-}4, 1e{-}4, 3e{-}5, 1e{-}5, 3e{-}6\}$. We tune over $\beta_2 \in \{0.9, 0.99\}$. We set the perturbation value $\xi = 10^{-10}$, following the default values in TensorFlow, except for the experiments in Section 5.6.1. In Section 5.6.1, we show the effect on convergence and generalization properties of ADAM and RMSProp when changing this parameter ξ.

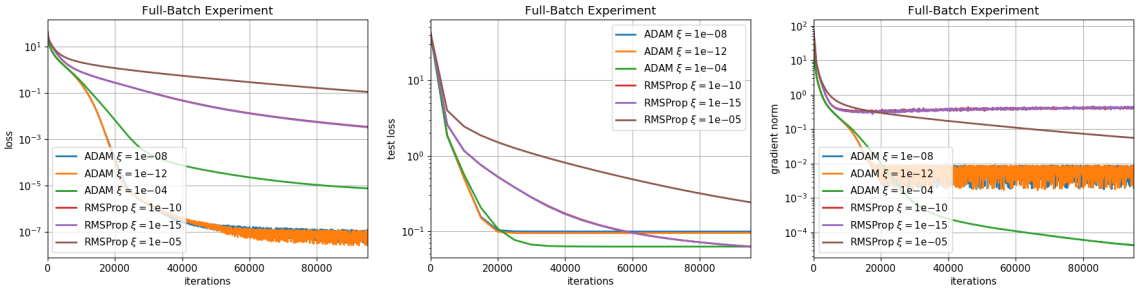
Note that ADAM and RMSProp uses an accumulator for keeping track of decayed squared gradient $\mathbf{v}_t$. For ADAM this is recommended to be initialized at $\mathbf{v}_0 = 0$. However, we found in the TensorFlow implementation of RMSProp that it sets $\mathbf{v}_0 = \mathbf{1}_d$. Instead of using this version of the algorithm, we used a modified version where we set $\mathbf{v}_0 = 0$. We typically found setting $\mathbf{v}_0 = 0$ to lead to faster convergence in our experiments.

ADAM The initial set of step sizes used were: $\{3e{-}4, 1e{-}4, 3e{-}5, 1e{-}5, 3e{-}6\}$. For ADAM, we tune over β_1 values of $\{0.9, 0.99\}$. For ADAM, We set $\beta_2 = 0.999$ for all our experiments as is set as the default in TensorFlow. Unless otherwise specified we use for the perturbation value $\xi = 10^{-8}$ for ADAM, following the default values in TensorFlow.

Contrary to what is the often used values of β_1 for ADAM (usually set to 0.9), we found that we often got better results on the autoencoder problem when setting $\beta_1 = 0.99$.

5.E Effect of the ξ parameter on adaptive gradient algorithms

In Figure 5.E.1, we show the same effect of changing ξ as in Section 5.6.1 on a 1 hidden layer network of 1000 nodes, while keeping all other hyper parameters fixed (such as learning rate, β_1, β_2). These other hyper-parameter values were fixed at the best values of these parameters for the default values of ξ, i.e., $\xi = 10^{-10}$ for RMSProp and $\xi = 10^{-8}$ for ADAM.



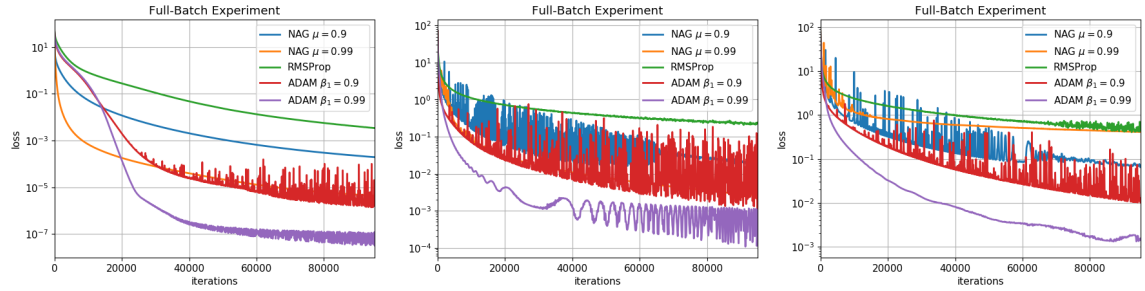
(A) Loss on training set (B) Loss on test set (C) Gradient norm on training set

FIGURE 5.E.1: Fixed parameters with changing ξ values. 1 hidden layer network of 1000 nodes

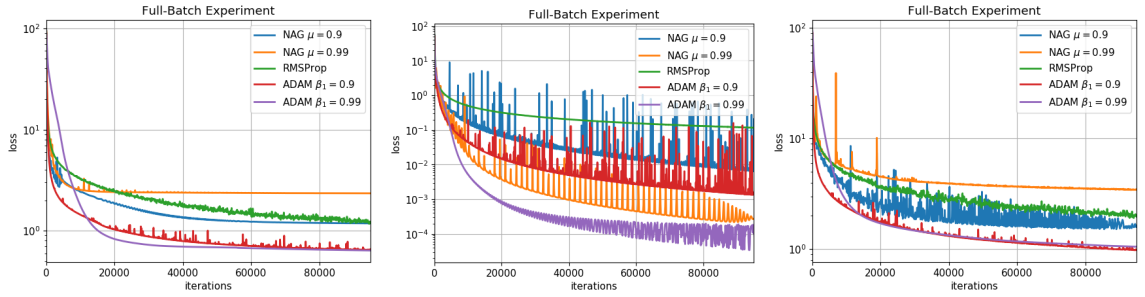
5.F Additional Experiments

5.F.1 Additional full-batch experiments on 22×22 sized images

In Figures 5.F.1, 5.F.2 and 5.F.3, we show training loss, test loss and gradient norm results for a variety of additional network architectures. Across almost all network architectures, our main results remain consistent. ADAM with $\beta_1 = 0.99$ consistently reaches lower training loss values as well as better generalization than NAG.

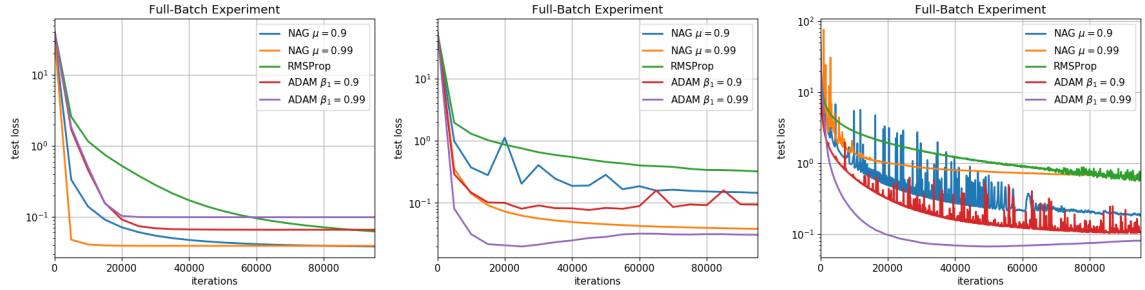


(A) 1 hidden layer; 1000 nodes (B) 3 hidden layers; 1000 nodes each (C) 5 hidden layers; 1000 nodes each

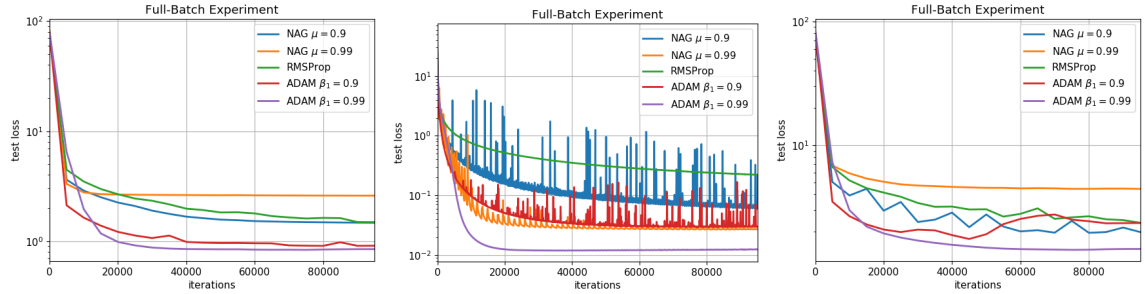


(D) 3 hidden layers; 300 nodes (E) 3 hidden layer; 3000 nodes (F) 5 hidden layer; 300 nodes

FIGURE 5.F.1: Loss on training set; Input image size 22×22

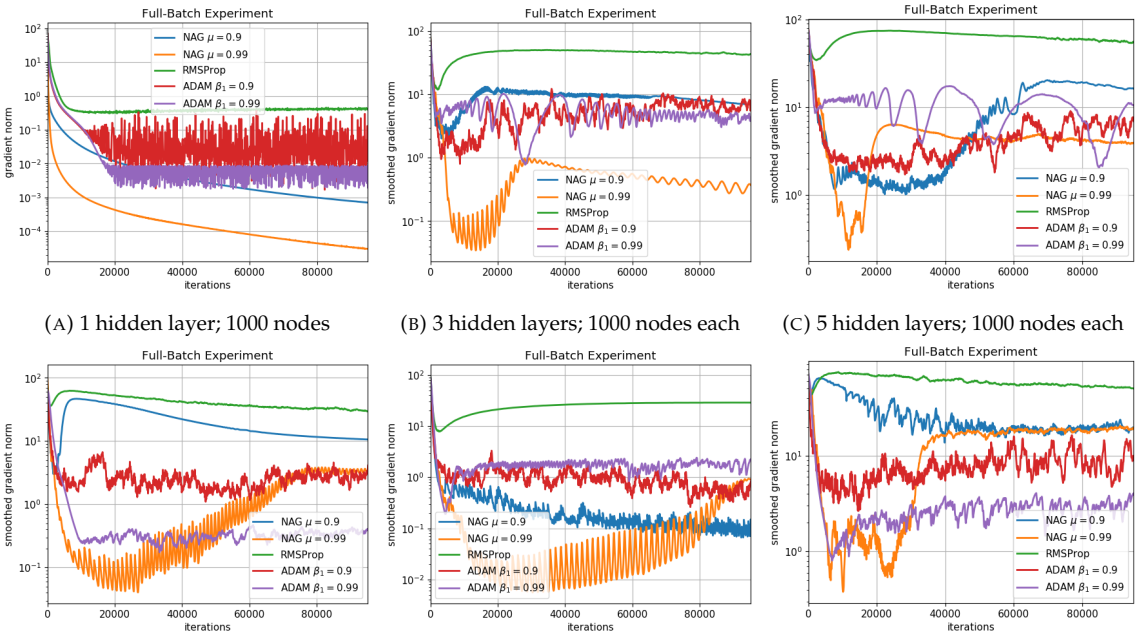


(A) 1 hidden layer; 1000 nodes (B) 3 hidden layers; 1000 nodes each (C) 5 hidden layers; 1000 nodes each



(D) 3 hidden layers; 300 nodes (E) 3 hidden layer; 3000 nodes (F) 5 hidden layer; 300 nodes

FIGURE 5.F.2: Loss on test set; Input image size 22×22



(A) 1 hidden layer; 1000 nodes

(B) 3 hidden layers; 1000 nodes each

(C) 5 hidden layers; 1000 nodes each

(D) 3 hidden layers; 300 nodes

(E) 3 hidden layer; 3000 nodes

(F) 5 hidden layer; 300 nodes

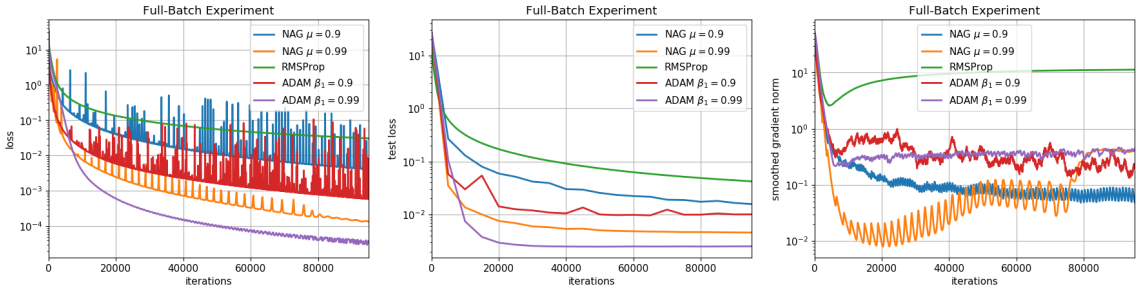
FIGURE 5.F.3: Norm of gradient on training set; Input image size 22×22

5.F.2 Are the full-batch results consistent across different input dimensions?

To test whether our conclusions are consistent across different input dimensions, we do two experiments where we resize the 22×22 MNIST image to 17×17 and to 12×12. Resizing is done using TensorFlow's `tf.image.resize_images` method, which uses bilinear interpolation.

Input images of size 17×17

Figure 5.F.4 shows results on input images of size 17×17 on a 3 layer network with 1000 hidden nodes in each layer. Our main results extend to this input dimension, where we see ADAM with $\beta_1 = 0.99$ both converging the fastest as well as generalizing the best, while NAG does better than ADAM with $\beta_1 = 0.9$.

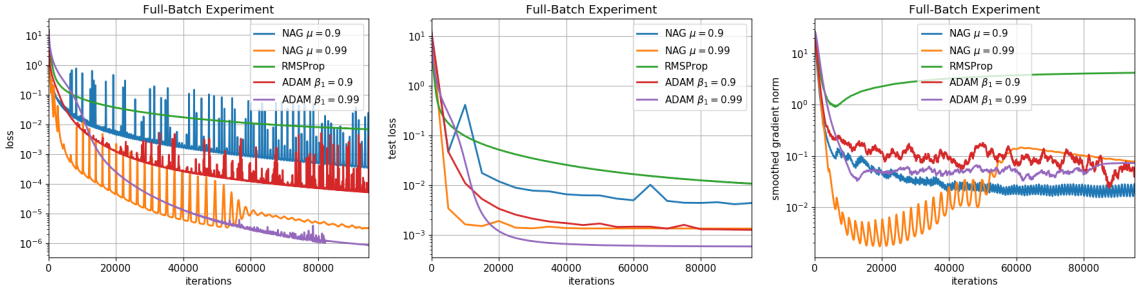


(A) Training loss (B) Test loss (C) Gradient norm

FIGURE 5.F.4: Full-batch experiments with input image size 17×17

Input images of size 12×12

Figure 5.F.5 shows results on input images of size 12×12 on a 3 layer network with 1000 hidden nodes in each layer. Our main results extend to this input dimension as well. ADAM with $\beta_1 = 0.99$ converges the fastest as well as generalizes the best, while NAG does better than ADAM with $\beta_1 = 0.9$.

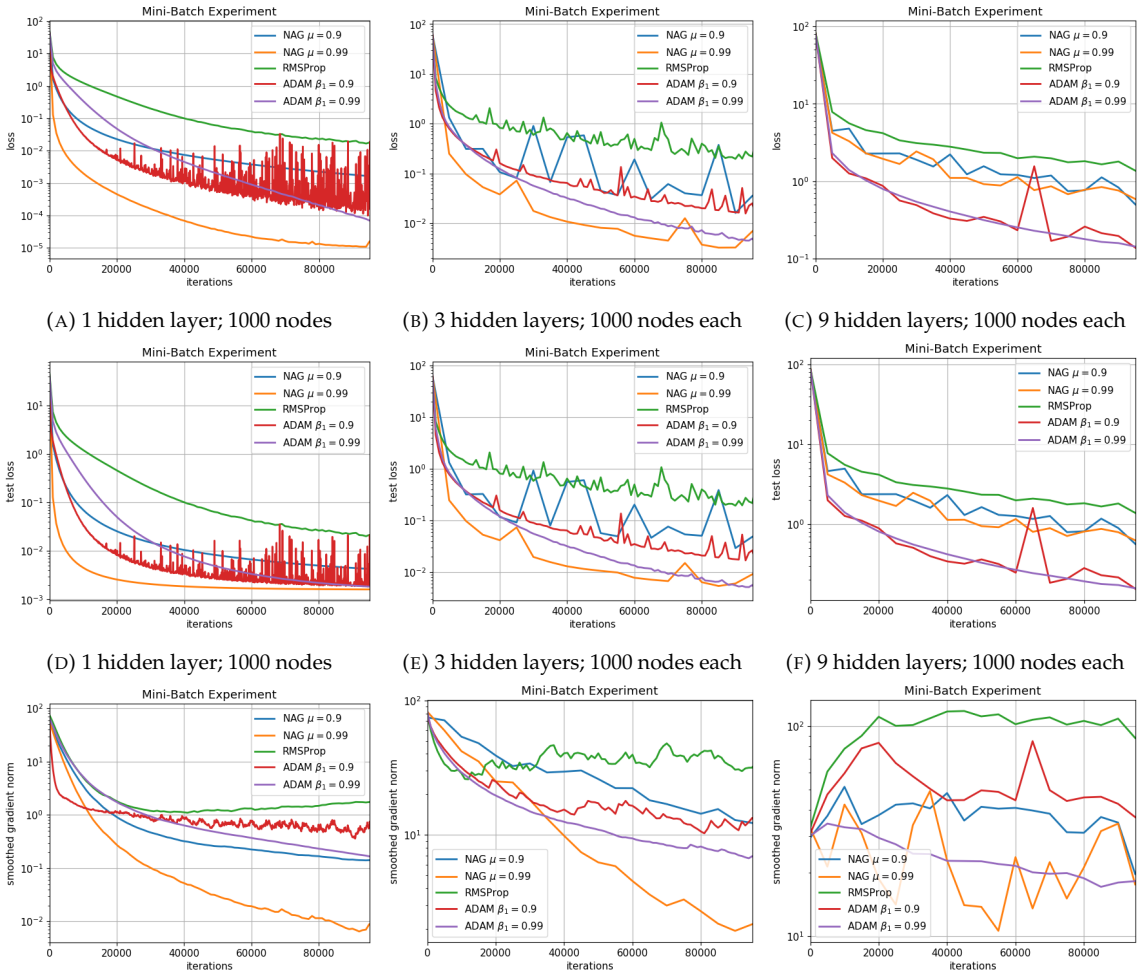


(A) Training loss (B) Test loss (C) Gradient norm

FIGURE 5.F.5: Full-batch experiments with input image size 12×12

5.F.3 Additional mini-batch experiments on 22×22 sized images

In Figure 5.F.6, we present results on additional neural net architectures on mini-batches of size 100 with an input dimension of 22×22. We see that most of our full-batch results extend to the mini-batch case.



(A) 1 hidden layer; 1000 nodes (B) 3 hidden layers; 1000 nodes each (C) 9 hidden layers; 1000 nodes each

(D) 1 hidden layer; 1000 nodes (E) 3 hidden layers; 1000 nodes each (F) 9 hidden layers; 1000 nodes each

(G) 1 hidden layer; 1000 nodes (H) 3 hidden layers; 1000 nodes each (I) 9 hidden layers; 1000 nodes each

FIGURE 5.F.6: Experiments on various networks with mini-batch size 100 on full MNIST dataset with input image size 22×22. First row shows the loss on the full training set, middle row shows the loss on the test set, and bottom row shows the norm of the gradient on the training set.

Chapter 6

PAC-Bayesian Risk Bounds for Neural Nets

6.1 Introduction

At the end of the thesis we finally arrive to explore what is possibly the deepest and the hardest question about neural nets and that is to understand their risk function. A long standing open question in deep-learning is to be able to theoretically explain as to when and why do neural nets which are massively over-parameterized happen to also minimize the risk even when they fit the training data arbitrarily accurately. Attempts to explain this have led to obtaining of risk bounds which do not scale with the number of parameters being trained on, (Bartlett, 1998; Golowich, Rakhlin, and Shamir, 2018; Harvey, Liaw, and Mehrabian, 2017). Recently it has been increasingly realized that good risk bounds can be obtained by making the bounds more sensitive to the training algorithm as well as the training data,(Arora et al., 2019a).

The range of available methods to bound risk or generalization error have been beautifully reviewed in Audibert and Bousquet, 2007. Here the authors have grouped the techniques into primarily four categories, (1) "Supremum Bounds" (like generic chaining, Dudley integral, Rademacher complexity), (2) "'Variance Localized Bounds", (3) "Data-Dependent Bounds" and (4) "Algorithm Dependent Complexity". The last category includes PAC-Bayes bounds which have resurfaced as a prominent candidate for a framework to understand risk of neural nets.

Over the years PAC-Bayesian risk bounds have been formulated in many different forms, (Hinton and Van Camp, 1993; McAllester, 1999; Langford and Seeger, 2001; McAllester, 2003). In the last couple of years, works like Dziugaite and Roy, 2017, Dziugaite and Roy, 2018a, Dziugaite and Roy, 2018b and Zhou et al., 2018b have shown the power of the PAC-Bayesian form of analysis of risk of neural nets. To the best of our knowledge "computational" bounds as demonstrated in the above reference are the first examples of non-vacuous/non-trivial upperbounds for risks of neural nets of any kind.

The above bounds are "computational" in the sense that they are obtained as outputs of an algorithmic search over hyperparameters on which the posterior distribution used in the bound depends on. These experiments strongly motivate the current work to search for stronger theoretical basis towards explaining the power of PAC-Bayesian risk bounds in explaining the generalization ability of neural nets. We make progress by identifying certain geometrical properties of the process of training nets which can be leveraged into getting better risk bounds.

6.1.1 A summary of our contributions

In works like Nagarajan and Kolter, 2019b, Nagarajan and Kolter, 2019a, it has been previously understood that risk bound on nets get better if they can appropriately utilize the information about the distance of the trained net from initialization. In this work we take a more careful look at this idea. We can decompose the distance from initialization into two independent quantities (a) a non-compact part, the change in the norm of the vector of weights of the net (i.e the sum of Frobenius norms of the layer matrices for a net without bias weights) and (b) a compact part, the angular deflection of this vector of weights from initialization to the end of training. Previous PAC-Bayes bounds have used data dependent priors to track the geometric mean of the spectral norms of the layer matrices (denoted as β) and have thus tracked the first parameter above. In this work we propose a mechanism of choosing priors in a data-dependent way from a two-indexed finite set, tracking both the quantities specified above. The compact angle is tracked by the index called λ in our two-indexed set of priors as specified in Definition 26.

Our second key innovation is that we show how in the PAC-Bayesian framework one can leverage more out of the angle parameter by simultaneously training a cluster of nets. In our risk bound in Theorem 6.3.1 **(the main theorem)** we imagine starting from a net f_{B} to get to the trained net f_{A} - the bold faced letter in the subscript of f will denote the (very high dimensional) vector of weights of the nets.

But alongside training f_{B} to obtain f_{A}, we also obtain a set $\{f_{\mathrm{A}_i}\}_{i=1,\ldots,k_1}$ of trained nets - of the same architecture as f_{A} and obtained using the same data set and using the same algorithm as was used to obtain f_{A}. This cluster of k_1 nets are obtained by doing training starting from multiple instances of weights initialized at different weight vectors, $\{\mathrm{B}_{\lambda^*,j}\}_{j=1,\ldots,k_1}$, of the *same norm* as B but within a cone around B whose half-angle λ^* is determined in a data-dependent way. The angle index λ of the set of priors, that we introduced previously, covers this choice of the conical half-angle.

Because of this use of clusters, compared to previous bounds our dependency on the distance from initialization is also more intricate. Our risk bound as derived in Theorem 6.3.1 can be seen to be

scaling with an effective notion of distance between any one of the A_i and the *initial cluster* of weights around B the $\{\mathrm{B}_{\lambda^*,j}\}_{j=1,\ldots,k_1}$. The bound has the flexibility that it will allow us to choose the A_i which has the smallest value of this effective distance and thus we are able to be more sensitive to the average behaviour. That is, for h being the width of the depth d nets being used, if the i^{th} net of the *final cluster* $\{\mathrm{A}_j\}_{j=1,\ldots,k_1}$ is closest to the *initial cluster* $\{\mathrm{B}_{\lambda^*,j}\}_{j=1,\ldots,k_1}$ then a crude "order" estimate of the risk bound on the stochastic net centered at f_{A}, that is given by Theorem 6.3.1 can be written as,

$$\mathbf{O}\left(\frac{\sqrt{h\log\left(\frac{2dh}{\delta}\right)}}{\sqrt{\text{training set size}}}\times d\cdot B\Big(\prod_{\ell=1}^{d}\|\mathrm{A}_{i,\ell}\|_2\Big)^{1-\frac{1}{d}}\times\frac{\text{inter-cluster distance between } \mathrm{B}_{\lambda^*,j}s \text{ \& } \mathrm{A}_j s}{\gamma}\right)$$

In above, B is a bound on the input vectors at training, the $\ell^{th}-$layer matrix corresponding to A_i is denoted as $\mathrm{A}_{i,\ell}$, γ is the margin value of which the margin-loss is being evaluated and the failure probability for the bound to hold is $\mathcal{O}(\delta)$. The exact formula for the bound given in Theorem 6.3.1 **(the main theorem)** makes it explicit that we have effectively built into the theory more data-dependent (and hence tunable) parameters which help us improve over existing bounds in multiple conceptual ways.

We would like to emphasize that our ability to exploit the cluster construction is crucially hinged on us being able to prove novel data-dependent noise resilience theorems for neural nets as given in Theorem 6.2.1. This theorem is potentially of independent interest and forms the technical core of our theoretical contribution. To the best of our knowledge this is the first such construction of a multi-parameter family of noise distributions on the weights of a neural net with guarantees of stability. In other words, if the net's weights are sampled from any of these noise distributions constructed in the theorem then w.h.p the output on the given data-set is guaranteed to not deviate too much from the given neural function on that architecture.

Summary of the experimental evidence in favour of our bounds We choose to compare our results against the bounds from Neyshabur et al., 2017 which we have in turn restated in subsection 6.1.2 with more accurate tracking of the various parameters therein. There are two ways to see why they are the appropriate baseline for comparison. (a) *Firstly* since our primary goal is to advance mathematical techniques to get better PAC-Bayesian risk bounds on nets we want to compare to other *theoretical* bounds in the same framework. The result from Neyshabur et al., 2017 are known to be the current state-of-the-art PAC-Bayesian risk bounds on nets. (b) *Secondly* to the best of our knowledge, for the range of depths of neural architectures that we are experimenting on, the bounds in

Neyshabur et al., 2017 are the state-of-the-art among all risk bounds (PAC-Bayesian or otherwise) on nets as has been alluded to in Nagarajan and Kolter, 2019a.

Further when two different theoretical expressions for risk bounds on nets depend on different sets of parameters of the neural net and its training process, we have to rely on empirical comparison. We recall that under similar situation this was also the adopted method of comparison to baselines for the state-of-the-art compression techniques of risk bounds in Arora et al., 2018.

In the experiments in Section 6.4 we will show multiple instances of nets trained over synthetic data and CIFAR-10 where we supersede the existing state-of-the-art in theoretical PAC-Bayesian bounds of Neyshabur et al., 2017. On these two datasets we probe very different regimes of neural net training in terms of the typical values of the angular deflection seen when obtaining the trained weights A from the initial weights B. In both these situations our bounds are lower than those from, Neyshabur et al., 2017 as we increase both the depth and the width. *Since we plot the bounds in the log scale for comparison we can conclude from the figures that not only do we have lower/tighter upperbounds we indeed also have better "rates" of dependencies on the architectural parameters.*

In the experiments we also demonstrate different properties of the path travelled by the neural net's weight vector during training and these we report in Section 6.5. For instance we observe that the $2-$norm of the weight vector of the net increases during training by a multiplicative factor which varies very little across all the experiments. The factor is between 1 and 3 and is fairly stable for one order of magnitude change in depths.

6.1.2 Reformulation of the PAC-Bayesian risk bound on neural nets from, Neyshabur et al., 2017

We start from Theorem 31.1 in Shalev-Shwartz and Ben-David, 2014 on PAC-Bayesian risk bounds which we re-state below as Theorem 6.1.1.

Definition 21. Let $\mathcal{H}$ be a hypothesis class, let $h \in \mathcal{H}$, let $\mathcal{D}$ be a distribution on an instance space Z, let $\ell : \mathcal{H} \times Z \to [0,1]$ be a loss function and let S be a finite subset of Z. Let m denote the size of S, i.e. the training-set size. $z \sim \mathcal{D}$ denotes sampling z from $\mathcal{D}$ and with slight abuse of notation $z \sim S$ denotes sampling z from a uniform distribution over S whenever S is a finite set. Further we define the expected and empirical risks for h as

$$L(h) := \mathbb{E}_{z\sim\mathcal{D}}[\ell(h,z)] \text{ and } \hat{L}(h) := \mathbb{E}_{z\sim S}[\ell(h,z)]$$

Theorem 6.1.1. (PAC-Bayesian Bound On Risk) Consider being given $\mathcal{D}$, $\mathcal{H}$ and ℓ as defined above. Let $\mathcal{H}$ also be equipped with the structure of a probability space. Let P, Q be two distributions over $\mathcal{H}$ called the "prior" and the "posterior" distribution respectively. Let $S \sim \mathcal{D}^m(Z)$ and then for every $\delta \in (0,1)$ we have the following guarantee:

$$\mathbb{P}_S \left[\forall Q \; \mathbb{E}_{h\sim Q}[L(h)] \leq \mathbb{E}_{h\sim Q}[\hat{L}(h)] + \sqrt{\frac{\mathrm{KL}(Q||P) + \log \frac{m}{\delta}}{2(m-1)}} \right] \geq 1-\delta$$

The above theorem shows that given a finite sample S from Z, we can choose the posterior distribution Q as a function of S and the above bound on generalization error is still guaranteed to hold w.h.p. The mechanism of choosing this Q in such a data-dependent way is made critical by the trade-offs between keeping the expected empirical risk of Q low and the KL divergence between P and Q low.

The above theorem applies to a wide class of loss functions, but for getting risk bounds specific to neural nets, hence forth we will focus on the following setup of classification loss.

Definition 22 (Margin Risk of a Multiclass Classifier). Define $\chi \subseteq \mathbb{R}^n$ to be the input space. Let $k \geq 2$ be the number of classes and set $Z = \chi \times \{1, \ldots, k\}$ for the rest of the chapter. Let $f_{\mathbf{w}} : \chi \to \mathbb{R}^k$ be a $k-$class classifier parameterized by the weight vector $\mathbf{w}$ and by "$f(\mathbf{x})[y]$" we shall mean the y^{th} coordinate of the output of f when evaluated on $\mathbf{x}$. Let $\gamma > 0$ be the "margin" parameter and then the "$\gamma-$Margin Risk" of f is

$$L_\gamma(f) := \mathbb{P}_{(\mathbf{x},y)\sim\mathcal{D}} \left[f(\mathbf{x})[y] \leq \gamma + \max_{i \neq y} f(\mathbf{x})[i] \right].$$

Analogously $\hat{L}_\gamma(f)$ denotes the "$\gamma-$Empirical Margin Risk" of f computed on a finite sample $S \subset Z$. We will use $m := |S|$.

Given this we can now present Theorem 6.1.2 below which is a slight variant of Lemma 1 of Neyshabur et al., 2017 and for completeness we give its proof in Appendix 6.C.

Theorem 6.1.2 (A special case of the PAC-Bayesian bounds for the margin loss). We follow the notation from Definition 22. Let $\{f_{\mathbf{w}} \mid \mathbf{w} \in \mathrm{W}\}$ denote a hypothesis class of $k-$class classifiers, where W is a space of parameters. Let P be any distribution (the "data-independent prior") on the space W. Then for any $\gamma > 0, \mathbf{w} \in \mathrm{W}$, and finite sample $S \subset Z$, define the family $D_{\gamma,\mathbf{w},S}$ of distributions on W such that for any $\mu \in D_{\gamma,\mathbf{w},S}$, we have,

$$P_{\mathbf{w}'\sim\mu}\left[\max_{(\mathbf{x},\mathbf{y})\in S}\|f_{\mathbf{w}'}(\mathbf{x})-f_{\mathbf{w}}(\mathbf{x})\|_\infty<\frac{\gamma}{4}\right]\geq\frac{1}{2}$$

Then for any $\gamma>0$ and $\delta\in[0,1]$, the following holds

$$\mathbb{P}_S\Bigg[\forall\mathbf{w}\text{ and }\mu\in D_{\gamma,\mathbf{w},S}\ \exists\text{ distribution }\tilde{\mu}\text{ on W s.t.}$$
$$\mathbb{E}_{\tilde{\mathbf{w}}\sim\tilde{\mu}}[L_0(f_{\tilde{\mathbf{w}}})]\leq\hat{L}_{\frac{\gamma}{2}}(f_{\mathbf{w}})+\sqrt{\frac{\mathrm{KL}(\mu||P)+\log\frac{3m}{\delta}}{m-1}}\Bigg]\geq 1-\delta \tag{6.1}$$

The proof of the above in Appendix 6.C gives an explicit expression of the distribution $\tilde{\mu}$ in terms of μ and $\mathbf{w}$. As is usual in practice the PAC-Bayesian bound in Theorem 6.1.2 will typically be used on a predictor whose weights have been obtained after training on a data-set S. Now we give a restatement of the risk bound on neural nets that was presented in Neyshabur et al., 2017 and towards that we need the following definition,

Definition 23 (Multiclass Neural-Network classifier). Define $f_{\mathrm{A}}:\mathbb{R}^n\to\mathbb{R}^k$ to be a depth-d neural-network with maximum width h, whose ℓ^{th} layer has weight matrix A_ℓ.[1] The first $d-1$ layers of f_{A} use the RELU non-linear activation.[2] $\mathrm{A}:=[\mathrm{vec}(\mathrm{A}_1);\ldots;\mathrm{vec}(\mathrm{A}_d)]$ is the vector of parameters formed by concatenating vectorized layer matrices and each coordinate of A is a distinct trainable weight in the net. Let $\mathcal{N}_{(\mathrm{A},\sigma^2)}$ denote the isotropic multivariate Gaussian probability distribution with mean A and variance $\sigma^2\mathbf{I}$ Define $\beta(\mathrm{A})=\left(\prod_{\ell=1}^d\|A_\ell\|_2\right)^{1/d}$. We will omit the argument A whenever the neural-network under consideration is clear from the context. Clearly β^d upper bounds the Lipschitz-constant for f_{A}.

Using the definitions above, we now present Theorem 6.1.3 where we give a reformulation of the "spectrally-normalized margin bound" originally given in Neyshabur et al., 2017.

Theorem 6.1.3 (Spectrally-Normalized Margin bound). Let S and m be as defined in Definition 22 and B be a bound on the norm of the input space χ from Definition 22. Let $f_{\mathbf{w}}$ be a given neural net with parameters as above and let its layer matrices be $\{W_\ell\}_{\ell=1}^d$. Construct a grid $\mathcal{B}$, called the "beta-grid", containing $K=\frac{d}{2}\times\left(\frac{\sqrt{m-1}}{2\exp(3-2/(d-1))}\right)^{1/d}$ uniformly spaced points covering the interval

[1] This network does not have any bias weights.
[2] In general any 1-Lipschitz activation will do.

$\left[\left(\frac{\gamma}{2B}\right)^{1/d}, \left(\frac{\sqrt{m-1}\gamma}{4\exp(3-2/(d-1))B}\right)^{1/d}\right]$. If $\tilde{\beta} = \text{argmin}_{x\in\mathcal{B}} |x - \beta(\mathbf{w})|$ and

$$\sigma(\tilde{\beta}) := \frac{1}{d\sqrt{2h\log(4dh)}} \min\left\{\frac{\gamma}{4e^2 B\tilde{\beta}^{d-1}}, \frac{\tilde{\beta}}{e^{\frac{1}{d-1}}}\right\} \tag{6.2}$$

Then we have the following guarantee for all $\delta \in (0, \frac{1}{K})$,

$$\mathbb{P}_{S\sim\mathcal{D}^m}\left[\exists\, \tilde{\mu}_{\mathbf{w}} \text{ s.t } \mathbb{E}_{\mathbf{w}+\tilde{\mathbf{u}}\sim\tilde{\mu}_{\mathbf{w}}}[L_0(f_{\mathbf{w}+\tilde{\mathbf{u}}})] \leq \hat{L}_{\frac{\gamma}{2}}(f_{\mathbf{w}}) + \sqrt{\frac{1}{m-1}}\sqrt{\sum_{\ell=1}^{d}\frac{\|W_\ell\|_F^2}{2\sigma(\tilde{\beta})^2} + \log\frac{3m}{\delta}}\right] \geq 1 - K\delta \tag{6.3}$$

For completeness we have re-derived the above in Appendix 6.D.

Remark. Theorem 6.1.3 above slightly differs from the original statement of Theorem 1 in Neyshabur et al., 2017 because of the following adaptations and improvements that we have made, (a) we tracked the various constants more carefully, (b) we have removed some of their assumptions and have chosen to report the bound as being on a stochastic neural risk as is most natural in this context and (c) we used a more refined way to account for the data-dependent priors.

6.2 A noise resilience guarantee for a certain class of neural nets

Definition 24 (Mixture Parameters). Let $k_1 \geq 2$ denote the number of components in a "mixture" distribution. Let $\mathcal{A} = \{\mathrm{A}_i \in \mathbb{R}^{\dim(\mathrm{A})} \mid i = 1,..,k_1\}$ denote a set of neural net weight vectors on the underlying architecture of f_{A}. Let $\mathcal{P} = \{p_i \mid i = 1,\ldots,k_1\}$ be a set of non-zero scalars. The mixture weights satisfy $\sum_i p_i = 1$. Define $\beta_i := \beta(\mathrm{A}_i)$ as defined in Definition 23. Further define $\mathrm{A}_{i,\ell}$ to be the ℓ^{th} layer of A_i.

Using the above setup we can state our main technical result as follows,

Theorem 6.2.1 (Controlled output perturbation with noisy weights from a mixture of Gaussians). Given S and χ as in Definition 22, let $B > 0$ be s.t the input space χ is a subset of the ball of radius B around the origin in $\mathbb{R}^n$. Further let f_{A} and β be as in Definition 23 and $\mathcal{A}, \mathcal{P}$ and $\{\beta_i\}_{i=1,\ldots,k_1}$ as in Definition 24, we choose any $\epsilon > 0$ s.t the following inequalities hold,

$$\forall\, i\in\{1\ldots k_1\}, \mathbf{x}\in S\ \|f_{\mathrm{A}_i}(\mathbf{x}) - f_{\mathrm{A}}(\mathbf{x})\| \leq \epsilon\|f_{\mathrm{A}}(\mathbf{x})\|$$

Then for every $\gamma > \epsilon \max_{\mathbf{x}\in S}\|f_{\mathrm{A}}(\mathbf{x})\|$ and $\delta \in (0,1)$, we have,

$$\mathbb{P}_{A'\sim \mathrm{MG(posterior)}}\Big[\max_{\mathbf{x}\in S}\|f_{A'}(\mathbf{x}) - f_A(\mathbf{x})\| > 2\gamma\Big] \leq \delta$$

Where $\mathrm{MG(posterior)}(\mathbf{w}) = \sum_i p_i \mathcal{N}_{(\mathrm{A}_i,\sigma^2)}(\mathbf{w})$ and $\sigma \leq \frac{1}{\sqrt{2h\log\left(\frac{2dhk_1}{\delta}\right)}} \min_{1\leq i\leq k_1} \min\left\{\frac{\beta_i}{d}, \frac{\gamma}{k_1 edBp_i\beta_i^{d-1}}\right\}$.

The above theorem has been proven in Section 6.6.

6.3 Our PAC-Bayesian risk bound on neural nets

Now we use Theorem 6.2.1 about controlled perturbation of nets to write the following theorem which is adapted to the setting of the experiments to be described in Section 6.4. Towards that we define the following notion of a "nice" training data which captures the effect that a set of nets evaluates to almost the same output on some given dataset,

Definition 25 ("Nice" training dataset). Given neural weights A and $\{\mathrm{A}_i\}_{i=1,\ldots,k_1}$ as in Definition 24, we call a training dataset S as $(\epsilon,\gamma)-$nice w.r.t. them if it satisfies the following conditions:

1. $\max_{\mathbf{x}\in S}\|f_{\mathrm{A}_i}(\mathbf{x}) - f_{\mathrm{A}}(\mathbf{x})\| \leq \epsilon\|f_{\mathrm{A}}(\mathbf{x})\|, \forall 1\leq i\leq k_1$
2. $\gamma > \epsilon \max_{\mathbf{x}\in S}\|f_{\mathrm{A}}(\mathbf{x})\|$

Next we define a two-indexed set of priors which will be critical to our PAC-Bayesian bounds.

Definition 26 (Our $2-$indexed set of priors)**.** Let $B > 0$ be s.t the input space χ in Definition 22 is a subset of the ball of radius B around the origin in $\mathbb{R}^n$. Given S, m as in Definition 22 and d, h as in Definition 23, we choose scalars $d_{\min}, \gamma, \delta > 0$ s.t. the following interval I is non-empty,

$$I := \left[\left(\frac{\gamma}{2B}\right)^{\frac{1}{d}}, \left(\frac{\gamma\sqrt{2(m-1)}}{(8Be^3 dd_{\min})\sqrt{2h\log\left(\frac{2dh}{\delta}\right)}}\right)^{\frac{1}{(d-1)}}\right]$$

Let f_{B} be a neural network. Consider a finite set of indices $\Lambda = \{1,\ldots,314\}$. For each $\lambda \in \Lambda$ we are given k_1 distinct neural net weights $\{\mathrm{B}_{\lambda,j}\}_{j=1}^{k_1}$ within a conical half-angle of 0.01λ around B. For each λ, we construct a grid $\mathscr{B}_\lambda$, called the "beta-grid", containing at most,

$$K_1 = \frac{d}{2} \times \frac{\left(\frac{\gamma\sqrt{2(m-1)}}{(8Be^3 d d_{\min})\sqrt{2h\log\left(\frac{2dh}{\delta}\right)}} \right)^{\frac{1}{(d-1)}}}{\left(\frac{\gamma}{2B}\right)^{\frac{1}{d}}}$$

points inside the interval I specified above. Now for each $\lambda \in \Lambda$ and $\tilde{\sigma} \in \mathscr{B}_\lambda$ we consider the following mixture of Gaussians $\frac{1}{k_1}\sum_{j=1}^{k_1} \mathcal{N}_{(\mathrm{B}_{\lambda,j}, \tilde{\sigma}^2 I)}$. Thus we have a grid of priors of total size $K := 314K_1$.

Remark. (a) Note that this set of mixture of Gaussian priors above indexed by λ and $\tilde{\sigma}$ corresponds to the set of distributions that we call $\{\pi_i\}$ in the general Theorem 6.B.1 (b) The specific choice of the set Λ given above is only for concreteness and to keep the setup identical to the experiments in sections 6.4 and 6.5 and this choice is *not* crucial to the the main theorem to be given next.

The choice of the parameters $\Lambda, \gamma, \epsilon$ and $d_{\min}$ The parameter B gets fixed by the assumption of boundedness of the data space χ and we choose the training data size m. The set Λ above is a convenient choice that we make motivated by experiments as a way to index a grid on the $\pi-$radians of possible deflection that can happen when the net f_{B} is trained to some final net (which we have been denoting as f_{A} as in subsection 6.1.1). Also note that in practice when presented with the neural nets f_{A} and $\{f_{\mathrm{A}_i}\}_{i=1}^{k_1}$ (which in turn fixes the value of depth d and width h) and the training data set S we would choose the smallest values of γ and ϵ so that the conditions in Definition 25 are satisfied. Then we choose δ (typically $\delta = 0.05$) which determines our confidence parameter $1-\delta$. At this point except $d_{\min}$ all other parameters are fixed that go into determining the interval I above. Now we can just choose $d_{\min}$ low enough so that the interval I is non-empty. Once $d_{\min}$ is chosen the value of K_1 and hence the size of the prior set also gets fixed.

Now we use the above definitions and the notations therein to state our main theorem as follows,

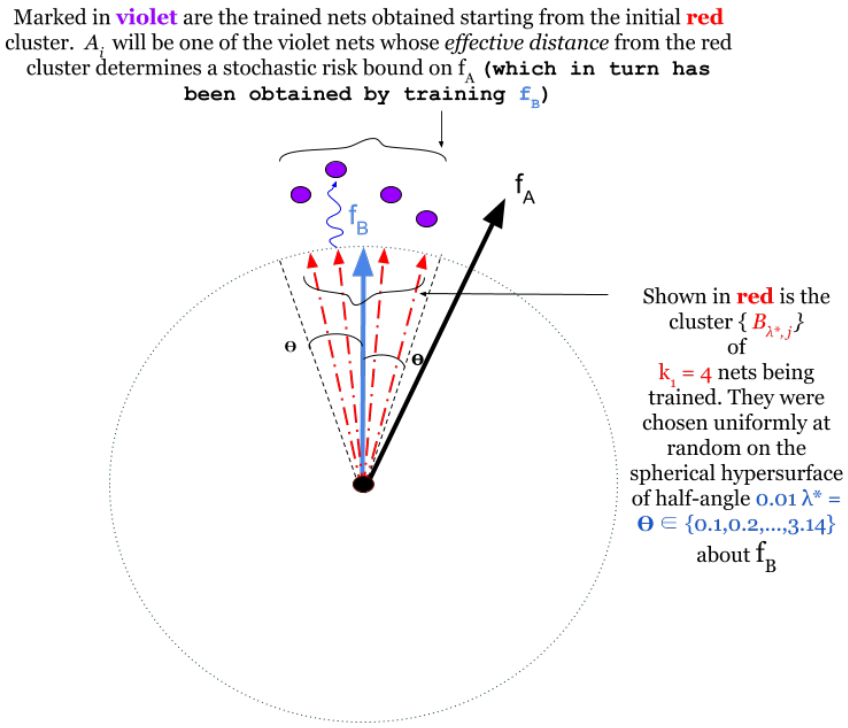


FIGURE 6.3.1: Starting from the weight vector B we get the trained weight vector A. Θ is the angle to which the angle of deflection $\measuredangle(A, B)$ has been discretized to.

Theorem 6.3.1 (Gaussian-Mixture PAC-Bayesian Bound). As indicated in Figure 6.3.1, suppose we train using the dataset S to obtain the trained net f_A from an initial neural net f_B. Let $\alpha = \arccos \frac{\langle A,B\rangle}{\|A\|\|B\|}$. Let $\lambda^* = \text{argmin}_{\lambda\in\Lambda} |0.01\lambda - \alpha|$. Further, let the neural weight vectors $\{A_i\}_{i=1,\ldots,k_1}$ be obtained by training the nets $\{f_{B_{\lambda^*,j}}\}_{j=1,\ldots,k_1}$ on S. Further for each such i define $d_{i,*} = \min_{j=1,\ldots,k_1} \|A_i - B_{\lambda^*,j}\|^2$ and $\tilde{\beta}_i = \text{argmin}_{x\in\mathcal{B}_{\lambda^*}} |x - \beta(A_i)|$.

Then it follows that for all $\epsilon > 0$ and $\delta \in (0, \frac{1}{K})$,

$$\mathbb{P}_S \Bigg[\exists\, \tilde{\mu}_A \text{ s.t. }, \forall i \text{ s.t. } d_{i,*} \geq d_{\min}, \mathbb{E}_{A+\tilde{u}\sim\tilde{\mu}_A}[L_0(f_{A+\tilde{u}})] \leq \hat{L}_{\frac{\gamma}{2}}(f_A) + \sqrt{\frac{1}{m-1}\sqrt{-\log\left(\frac{1}{k_1}\sum_{j=1}^{k_1}\exp(-\frac{1}{2\tilde{\sigma}_i^2}\|A_i - B_{\lambda^*,j}\|^2)\right) + \log\frac{3m}{\delta}}} \;\Big|\; S \text{ is } (\epsilon,\gamma)\text{-nice w.r.t } \{A, A_{i=1,\ldots,k_1}\} \Bigg] \geq 1 - K\delta \tag{6.4}$$

where $\tilde{\sigma}_i^2 := \frac{1}{2h\log\left(\frac{2dh}{\delta}\right)} \left(\min\left\{\frac{\tilde{\beta}_i}{de^{\frac{1}{d-1}}}, \frac{\frac{\gamma}{8}}{e^2 dB\tilde{\beta}_i^{d-1}}\right\}\right)^2$

The proof of the above Theorem has been given in Section 6.7

Remark. We emphasize that the structure of the above theorem is the same as that of the general PAC-Bayes bound as stated in Theorem 6.1.1. The distribution $\tilde{\mu}_{\mathrm{A}}$ above is a choice of the posterior distribution that is called Q in the general theorem. Hence its w.r.t this $\tilde{\mu}_{\mathrm{A}}$ that we are bounding with high probability the stochastic risk of the neural net f_{A} under the loss function L_0

Secondly the distribution $\tilde{\mu}_{\mathrm{A}}$ here is explicitly constructed such that, for the sampled noisy weights $\mathrm{A} + \tilde{\mathbf{u}}$ it is ensured that $\max_{\mathbf{x}\in S}\|f_{\mathrm{A}+\tilde{\mathbf{u}}}(\mathbf{x}) - f_{\mathrm{A}}(\mathbf{x})\|_\infty < \frac{\gamma}{4}$. Also we emphasize that in the above λ^* as defined is s.t $0.01 \times \lambda^*$ is the closest angle in the set $\{0.01, 0.02, \ldots, 3.14\}$ to the angle between the initial net's weight vector B and the trained net's weight vector A

We note the following salient points about the above theorem,

- In our experiments the nets $\{\mathrm{A}_i\}_{i=1,\ldots,k_1}$ will be obtained by training the nets $\{f_{\mathrm{B}_{\lambda^*,j}}\}_{j=1,\ldots,k_1}$ on the same training data S using the same method by which we obtain f_{A} from f_{B}.

- We emphasize that the above setup is not tied to any specific method of obtaining the initial and final clusters of nets. For example there are many successful heuristics known for "compressing" a neural net while approximately preserving the function. One can envisage using the above theorem when such a heuristic is used to get such a cluster $\{B_{\lambda^*,j}\}$ from B and $\{\mathrm{A}_i\}$ from A.

- For each of the different choices amongst $\{\mathrm{A}_i\}_{i=1}^{k_1}$ which satisfy the condition $d_{i,*} \geq d_{\min}$ we can get a different upper bound on the risk in the LHS. Hence this theorem gives us the flexibility to choose amongst $\{\mathrm{A}_i\}_{i=1}^{k_1}$ those that give the best bound.

- We note that after a training set S has been sampled, we require no niceness condition on the nets which will have to hold over the entire domain of the nets. The upperbound as well as the confidence on the upperbound are all entirely determined by the chosen data-set S.

- Corresponding to the experiments in the next section we will observe in section 6.5 that the angular deflection α above is predominantly determined by the data-distribution from which S is being sampled from and at a fixed width it decreases slightly with increasing depth.

  The improvements seen in our approach to PAC-Bayesian risk bounds strongly suggest that the consistent patterns of dilation of the weight matrix norms (also reported in section 6.5) and the angular deflection of the net's weight vector merit further investigation.

6.4 Experimental comparison of our bounds with Neyshabur et al., 2017

Here we present empirical comparisons between OUR PAC-Bayesian risk bound on nets in Theorem 6.3.1 and the result in Theorem 6.1.3 reproduced from Neyshabur et al., 2017, which we denote as the NBS bound.

We compute the two bounds for neural nets without bias weights (as needed by these two theorems stated above). We posit that the fair way to do comparison is to only choose the initial net, data-set and the training algorithm (including a stopping criteria) and to compute the different theories on whatever is the net obtained at the end of the training. We test the theories on the following different classification tasks (a) binary classification tasks on parametrically varied kinds of synthetic datasets which have two linearly separable clusters and (b) multi-class classification of the CIFAR$-$10 dataset. In both the cases we study effects of varying the net's width and depth.

It is to be noted that all the following experiments have been also done over varying sizes of the training data set and the advantage displayed here of of OUR bound over NBS's result, is robust to this change.

6.4.1 CIFAR-10 Experiments

Here the nets we train are of depths $2, 3, \ldots, 8$ and 16 and we vary the number of ReLU gates in a hidden layer (h) between 100 and 200. We train the networks to a test-accuracy of approximately 50% which is close to the best known performance of feed-forward networks on the CIFAR-10 dataset. The neural networks are initialized using the "Glorot Uniform" initialization and we use the ADAM weight update rule on the cross-entropy training loss. In each epoch we use mini-batch size 300 and we set $k_1 = 25$ (k_1 as defined in Theorem 6.3.1).

Results We test both the theories, OUR bound in equation (6.4) and the NBS bound in equation (6.3), at 95% confidence i.e at $K\delta = 0.05$ in the above referenced equations.

Having trained the initial cluster as needed in Theorem 6.3.1, we choose the smallest ϵ and γ that satisfy the "niceness" condition in Definition 25. In experiments we see that often (not always) this minimum γ needed to satisfy this condition increases with the depth of the net. *At any fixed architecture and dataset we evaluate both the theories at the same value of γ chosen as said above.* We repeat the experiment with 10 different random seeds (which changes the data-set, the initial cluster choices and the mini-batch sequence in the training).

In Figure 6.4.1 we see examples of how OUR bounds do better than NBS. We plot OUR bound for the i^{th} point in the final cluster (as defined in equation 6.4) that achieves the lowest bound. (We always start from taking a very small value as the choice of $d_{\min}$, as required in Definition 26, s.t in experiments the distance between the clusters was always bigger than that.) Note the log-scale in the $y-$axis in this figure and hence the relative advantage of our bound is a significant multiplicative factor which is *increasing* with depth. *And at large widths our bound seems to essentially flatten out in its depth dependence.*

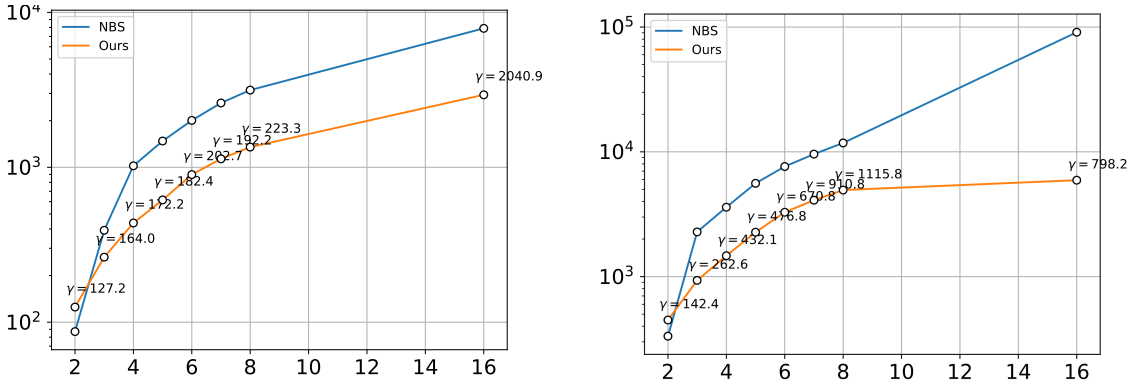


FIGURE 6.4.1: In the above figures we plot the risk bounds (in the $y-$axis) predicted by Theorem 6.3.1 and Theorem 6.1.3 for trained nets at different depths, the $x-$ axis. We can see the comparative advantage across depths in favour of OUR bound over NBS when tested on CIFAR-10 while the width of the net is 100 for the figure on the left and width is 400 for the figure on the right.

6.4.2 Synthetic Data Experiments

In this section we show a comparison between OUR and NBS bounds on a synthetic dataset which allows for probing the theories in a very different regime of parameters than CIFAR-10. Here the classification accuracies of the nets are near perfect, the margin parameters and the angular deflection of the net during training are significantly lower.

Dataset We randomly sample $m = 1000$ points in $\mathbb{R}^n$ from two different isotropic variance 1 normal distributions centered at $\mathbf{1} = (1, 1, \ldots, 1)$ and $a\mathbf{1}$ for $n = 20$ and 40 and for $a \in \{2, 4, 6, 8, 10\}$. We reject a sample $\mathbf{x}$ if $\min(||\mathbf{x} - \mathbf{1}||_\infty, ||\mathbf{x} - a\mathbf{1}||_\infty) > 1$. Thus the inter-cluster distance varies with a. When $n = 20$ then we set $B = 50$ and otherwise $B = 100$.

Architecture and training We train fully-connected feed-forward nets of depth $2, 3, \ldots, 8$ on the cross-entropy loss function. Each hidden layer has 800 ReLU gates. As before we initialize the neural network layers using the "Glorot Uniform" initialization and train using ADAM (mini-batch size 100). Networks with depth $d < 5$ required 5 epochs and $d \geq 5$ require 8 epochs for training to 100% train and test accuracy. Our risk bounds are computed using cluster size of $k_1 = 25$ as before.

Results The parameters K and δ are set so that the confidence on the bounds is at 95%. For each network depth we compute our bound 10 times using 10 different random seeds. Each trial achieves approximately 100% test accuracy and we plot the bound for the seed which achieves the minimum value of γ. For each network depth we label the value of γ used to compute the bounds. The same value of γ is used for computing both OUR and NBS bounds. We compute OUR bound for the i^{th} cluster point that achieves the lowest bound.

In Figure 6.4.2 we compare OUR bound in Theorem 6.3.1 to the NBS bound in Theorem 6.1.3 at two of the many parameter configurations of the above model where we have tested the theories. Again we find that our bound is consistently lower by a multiplicative factor than the previous PAC-Bayes bound.

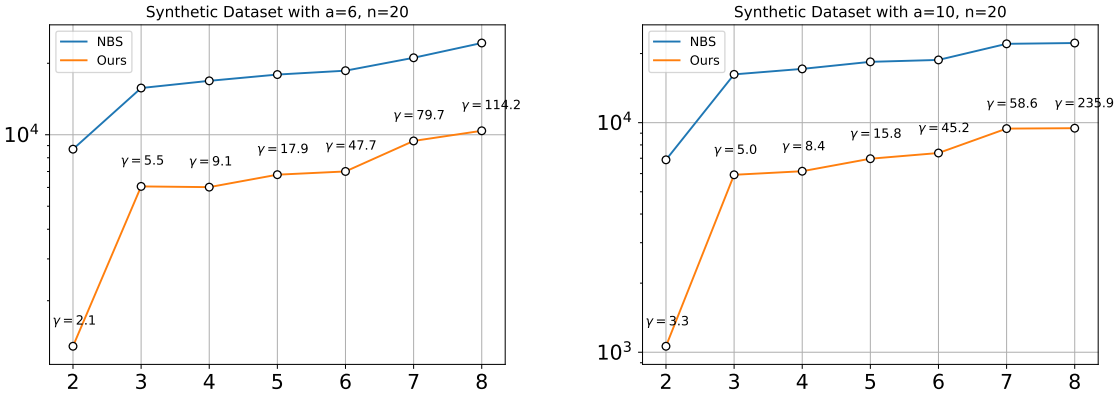


FIGURE 6.4.2: In the above figures we plot the risk bounds (in the $y-$axis) predicted by Theorem 6.3.1 and Theorem 6.1.3 for trained nets at different depths, the $x-$ axis. In particular here we compare the two theories when the synthetic data generation model is sampling in $n = 20$ dimensions with the cluster separation parameter $a = 6$ in the left figure and $a = 10$ in the right figure.

6.5 Experimental observations about the geometry of neural net's training path in weight space

Our approach to PAC-Bayesian risk bounds for neural nets motivates us to keep track as to how during training the norm of the neural net's weight vector changes and by how much angle this vector deflects. In here we record our observations about the interesting patterns that these two parameters were observed to have for the two kinds of experiments that were done in the previous section. The structured behaviours as seen here are potentially strong guidelines for directions for future theoretical developments.

Data from the experiments on CIFAR-10 in subsection 6.4.1

- We show in Figure 6.5.1 the Gaussian kernel density estimation, (Scott, 2015), of the angular deviation (α in Theorem 6.3.1) over the 10 trials at every depth at a width of 100. We can see that the angular deflection is fairly stable to architectural changes and is only slightly decreasing with depth. Similar pattern is also observed at higher widths.

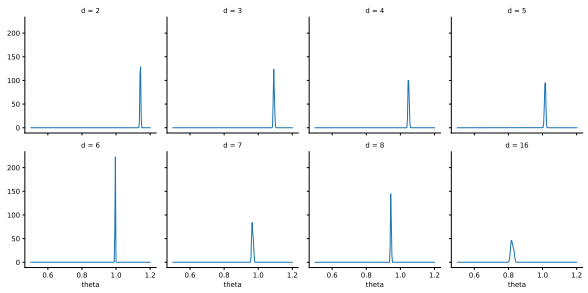


FIGURE 6.5.1: Gaussian kernel density estimate on 10 trials of the experiment (for every depth d and 100 width) measuring the angular deviation of the weight vector under training

- In Figure 6.5.2 we show the initial parameter norm vs the final parameter norm for training nets of different depths at width 100 on the CIFAR dataset. Thus its demonstrated that the multiplicative factor with which the norm increases during training is also fairly stable to architectural changes.

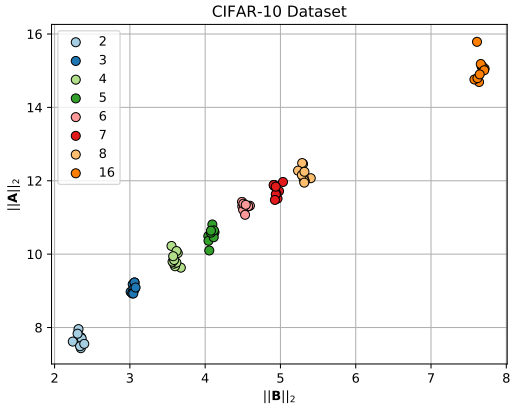


FIGURE 6.5.2: Initial parameter norm $\|B\|_2$ vs final parameter norm $\|A\|_2$ with increasing depths (at width 100) on the CIFAR-10 dataset. 10 trials are displayed for each architecture.

Data from the experiments on synthetic data in subsection 6.4.2

- In Figure 6.F.1 in Appendix 6.F we show the KDE of the angular deviation θ between the initial and final networks for different depths d and cluster separations a. The KDE was obtained from θ values obtained through 10 trials. Here we observe that the mean value of the angular deflection due to training is only slightly affected by the architectural choices or the cluster

separation parameter. The variance of the distribution of the angle increases with increase in d and a and the mean value tends to slightly decrease with increasing depths. (Note that the mean angular deflection for CIFAR-10 that we saw in the previous experiment was significantly larger than here.)

- Lastly in Figure 6.5.3 we show the variation between the initial and the final norms for the inter-cluster separation of $a = 2$. We observe a consistent (and surprising) behaviour that training seems to dilate the sum of Frobenius norms of the net and the dilation factor is close to 1 at depth 2 and increases to about 2.5 for about an order of magnitude of increase in depths. This behaviour is fairly stable across different values of a that we tried and recall that this same phenomenon was also demonstrated on CIFAR-10.

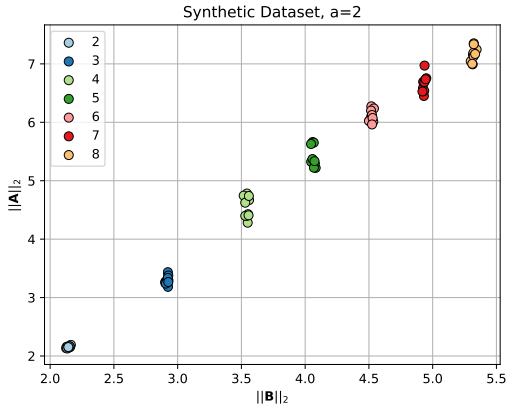


FIGURE 6.5.3: Scatter plot of $||A||_2$ versus $||B||_2$ for neural networks of depths $2, 3, \ldots, 8$, for cluster separation $a = 2$ in ambient dimension $n = 20$. For each depth we run 10 trials with different random initializations and mini-batch sequence in the SGD.

6.6 Proof of Theorem 6.2.1

Towards proving the main Theorem 6.2.1 we need the following definition and lemma,

Definition 27 (Random Neural Network). Let us denote as, $f_{\mathcal{N}_{(\mathrm{A}_i,\sigma^2)}}$ the random neural net function obtained by sampling its weights from the isotropic Gaussian distribution with p.d.f $\mathcal{N}_{(\mathrm{A}_i,\sigma^2)}$.

Lemma 6.6.1 (Controlled output perturbation of Gaussian weights). Let us be given a set of neural net weight vectors (for a fixed architecture) $\mathscr{A} = \{\mathrm{A}_i\}_{i=1,..,k_1}$ s.t that $\|\mathrm{A}_{i,\ell}\| = \beta_i$ for all $i \in \{1, \ldots, k_1\}$

and $\ell \in \{1, \ldots, d\}$. If,

$$\sigma \leq \frac{1}{\sqrt{2h \log \frac{2dhk_1}{\delta}}} \min_{i \in \{1,\ldots,k_1\}} \min \left\{ \frac{\beta_i}{d}, \frac{\gamma}{k_1 edBp_i \beta_i^{d-1}} \right\} \tag{6.5}$$

Then,

$$\mathbb{P}\left(\sum_{i=1}^{k_1} p_i \| f_{\mathcal{N}_{(\mathrm{A}_i, \sigma^2 I)}} - f_{\mathrm{A}_i} \| > \gamma \right) < \delta \tag{6.6}$$

Proof of Lemma 6.6.1. Let $\bar{U} = \{U_{i,l} \mid i = 1, \ldots, k_1, \ell = 1, \ldots, d\}$ be a set of size $d \cdot k_1$ containing $h-$dimensional random matrices, such that each matrix $U_{i,\ell} \sim \mathcal{N}(\mathbf{0}, \sigma^2)$

We can define matrices $\{\mathrm{B}_p \in \mathbb{R}^{h \times h} \mid p = 1, \ldots, h^2\}$ s.t each B_p has σ in an unique entry of it and all other entries are 0. Then it follows that as random matrices, $U_{i,\ell} = \sum_{p=1}^{h^2} \gamma_p \mathrm{B}_p$ with $\gamma_p \sim \mathcal{N}(0, 1)$. We note that $\| \sum_{p=1}^{h^2} \mathrm{B}_p \mathrm{B}_p^\top \| = h \cdot \sigma^2$ since h is the largest eigenvalue of an all ones $h-$dimensional square matrix. Now we invoke Corollary 4.2 of Tropp, 2012 here to get for any $t > 0$,

$$\mathbb{P}_{U_{i,\ell}} \left[\|U_{i,\ell}\|_2 > t \right] \leq 2he^{-\frac{t^2}{2h\sigma^2}} \tag{6.7}$$

Using union bound we have, $\mathbb{P}_{\bar{U}}[\exists (i, \ell) \text{ s.t } \|U_{i,\ell}\| > t_i] \leq 2dh \sum_{i=1}^{k_1} e^{-\frac{t_i^2}{2h\sigma^2}}$. So we have,

$$1 - 2dh \sum_{i=1}^{k_1} e^{-\frac{t_i^2}{2h\sigma^2}} \leq \mathbb{P}_{\bar{U}}[\forall (i, \ell) \text{ s.t } \|U_{i,\ell}\| \leq t_i] \tag{6.8}$$

Let $\mathrm{A}_{i,\ell}$ be the induced matrix in the $\ell^{th}-$layer from the neural weight vector A_i. Let $U_{i,l}$ be the perturbation for $\mathrm{A}_{i,l}$ and suppose, $\|U_{i,\ell}\| \leq \frac{1}{d} \|\mathrm{A}_{i,\ell}\|$.[3] We have by Lemma 2 of Neyshabur et al., 2017 and our assumption of uniform spectral norms for the layer matrices,

$$\| f_{\mathrm{A}_i + (\mathrm{vec}(U_{i,\ell}))_{\ell=1,2,\ldots,d}} - f_{\mathrm{A}_i} \| \leq eB\beta_i^{d-1} \sum_{\ell=1}^{d} \|U_{i,\ell}\|_2 \tag{6.9}$$

[3]Since the width of the net is assumed to be uniformly h it follows that $U_{i,\ell}$ is of the same dimensions as $\mathrm{A}_{i,\ell}$.

From equations (6.9-6.8) it follows that if $\forall i\, t_i \leq \frac{\beta_i}{d}$ then we have,

$$\begin{aligned}
1 - 2dh \sum_{i=1}^{k_1} e^{-\frac{t_i^2}{2h\sigma^2}} \leq & \mathbb{P}_{\bar{U}}[\forall (i,\ell) \text{ s.t } \|U_{i,\ell}\| \leq t_i] \\
\leq & \mathbb{P}_{\bar{U}}[\forall i, \|f_{\mathrm{A}_i + (\mathrm{vec}(U_{i,\ell}))_{\ell=1,2,..,d}} - f_{\mathrm{A}_i}\| \leq eB\beta_i^{d-1}(\sum_{l=1}^{d} t_i) = eB\beta_i^{d-1} dt_i] \\
\leq & \mathbb{P}\left[\sum_{i=1}^{k_1} p_i \|f_{\mathrm{A}_i + (\mathrm{vec}(U_{i,\ell}))_{\ell=1,2,..,d}} - f_{\mathrm{A}_i}\| \leq edB \sum_{i=1}^{k_1} p_i \beta_i^{d-1} t_i\right]
\end{aligned}$$

Let $\gamma > 0$ and chose t_i s.t

$$edB \sum_{i=1}^{k_1} p_i \beta_i^{d-1} t_i \leq \gamma \tag{6.10}$$

hence, we get

$$1 - 2dh \sum_{i=1}^{k_1} e^{-\frac{t_i^2}{2h\sigma^2}} \leq \mathbb{P}\left[\sum_{i=1}^{k_1} p_i \|f_{\mathrm{A}_i + (\mathrm{vec}(U_{i,\ell}))_{\ell=1,2,..,d}} - f_{\mathrm{A}_i}\| \leq \gamma\right] \tag{6.11}$$

A sufficient condition for (6.10) is $t_i \leq \frac{\gamma}{k_1 edB p_i \beta_i^{d-1}}$. Combined with the condition that $t_i \leq \frac{\beta_i}{d}$ it follows that (6.11) holds if $\forall i \in \{1, \ldots, k_1\}$,

$$t_i \leq \min\left\{\frac{\beta_i}{d}, \frac{\gamma}{k_1 edB p_i \beta_i^{d-1}}\right\} \tag{6.12}$$

To get (6.6) we choose σ s.t $\sum_{i=1}^{k_1} e^{-\frac{t_i^2}{2h\sigma^2}} \leq \frac{\delta}{2dh}$ This is ensured if $\max_{i=1,\ldots,k_1} e^{-\frac{t_i^2}{2h\sigma^2}} \leq \frac{\delta}{2dhk_1} \iff \frac{\min_{i=1,\ldots,k_1} t_i^2}{2h\sigma^2} \geq -\log\left(\frac{\delta}{2dhk_1}\right) \iff$

$$\sigma^2 \leq \frac{\min_{i=1,\ldots,k_1} t_i^2}{2h \log\left(\frac{2dhk_1}{\delta}\right)} \tag{6.13}$$

We can maximize σ^2 and obey constraints (6.13-6.12) by setting

$$\sigma^2 = \frac{1}{2h \log\left(\frac{2dhk_1}{\delta}\right)} \left(\min_{i \in \{1,\ldots,k_1\}} \min\left\{\frac{\beta_i}{d}, \frac{\gamma}{k_1 edB p_i \beta_i^{d-1}}\right\}\right)^2 \tag{6.14}$$

□

Using the above now we can demonstrate the proof of Theorem 6.2.1.

Proof of Theorem 6.2.1. Given the assumption we have that, $\forall i \in \{1, \ldots, k_1\}$ and $\mathbf{x} \in S$, $\|f_{\mathrm{A}_i}(\mathbf{x}) - f_{\mathrm{A}}(\mathbf{x})\| \leq \epsilon \|f_{\mathrm{A}}(\mathbf{x})\|$, we have the following inequality for all neural weights A',

$$\|f_{\mathrm{A}'}(\mathbf{x}) - f_{\mathrm{A}}(\mathbf{x})\| \leq \|f_{\mathrm{A}'}(\mathbf{x}) - f_{\mathrm{A}_i}(\mathbf{x})\| + \|f_{\mathrm{A}_i}(\mathbf{x}) - f_{\mathrm{A}}(\mathbf{x})\| \leq \|f_{\mathrm{A}'}(\mathbf{x}) - f_{\mathrm{A}_i}(\mathbf{x})\| + \epsilon\|f_{\mathrm{A}}(\mathbf{x})\| \tag{6.15}$$

$$\implies \|f_{\mathrm{A}'}(\mathbf{x}) - f_{\mathrm{A}}(\mathbf{x})\| \leq \epsilon\|f_{\mathrm{A}}(\mathbf{x})\| + \min_{i\in\{1,\ldots,k_1\}} \|f_{\mathrm{A}'}(\mathbf{x}) - f_{\mathrm{A}_i}(\mathbf{x})\| \tag{6.16}$$

$$\leq \epsilon\|f_{\mathrm{A}}(\mathbf{x})\| + Z \tag{6.17}$$

where in the last line above we have defined, $Z = \min_{i\in\{1,\ldots,k_1\}} \|f_{\mathrm{A}'}(\mathbf{x}) - f_{\mathrm{A}_i}(\mathbf{x})\|$. Now for a choice of $\gamma(\mathbf{x})$ s.t $\gamma(\mathbf{x}) > \epsilon\|f_{\mathrm{A}}(\mathbf{x})\|$ and using inequation 6.15 we have the following inequalities being true for the given distribution MG(posterior),

$$\mathbb{P}_{\mathrm{A}'\sim\text{MG(posterior)}} \left(\|f_{\mathrm{A}'}(\mathbf{x}) - f_{\mathrm{A}}(\mathbf{x})\| > 2\gamma(\mathbf{x})\right) \leq \mathbb{P}\left(\epsilon\|f_{\mathrm{A}}(\mathbf{x})\| + Z > 2\gamma(\mathbf{x})\right) \tag{6.18}$$

$$\leq \mathbb{P}\left(Z > 2\gamma(\mathbf{x}) - \epsilon\|f_{\mathrm{A}}(\mathbf{x})\|\right) \tag{6.19}$$

$$\leq \mathbb{P}\left(Z > \gamma(\mathbf{x})\right) \tag{6.20}$$

A' being randomly sampled from the distribution MG(posterior) can be imagined to be done in two distinct steps, (1) first we select a center among the set $\{\mathrm{A}_i\}_{i=1,\ldots,k_1}$ by sampling a random variable Y valued in the set, $\{1,\ldots,k_1\}$ with probabilities $\{p_i\}_{i=1,\ldots,k_1}$ and then (2) sample the weights from, $\mathcal{N}_{(\mathrm{A}_Y,\sigma^2 I)}$.

We define a collection of k_1 mutually independent random variables $\{Z_j := \min_{i\in\{1,\ldots,k_1\}} \|f_{\mathscr{P}}(\mathbf{x}) - f_{\mathrm{A}_i}(\mathbf{x})\|\}_{j=1,\ldots,k_1}$ with $\mathscr{P} \sim \mathcal{N}_{(\mathrm{A}_j,\sigma^2 I)}$. Clearly $Z_j = Z \mid Y = j$. Thus we have the following relationship among the events,

$$\{Z > \gamma(\mathbf{x})\} = \{\min_{i\in\{1,\ldots,k_1\}} \|f_{\mathrm{A}'}(\mathbf{x}) - f_{\mathrm{A}_i}(\mathbf{x})\| > \gamma(\mathbf{x})\} = \bigcup_{j=1}^{k_1} \{(Z > \gamma(\mathbf{x})) \cap (Y = j)\} \tag{6.21}$$

$$\implies \mathbb{P}\left(Z > \gamma(\mathbf{x})\right) = \sum_{j=1}^{k_1} \mathbb{P}\left(Z > \gamma(\mathbf{x}) \mid Y = j\right) \mathbb{P}\left(Y = j\right) = \sum_{j=1}^{k_1} \mathbb{P}\left(Z_j > \gamma(\mathbf{x})\right) \mathbb{P}\left(Y = j\right) \tag{6.22}$$

In the above we invoke the definition that, $\mathbb{P}(Y = j) = p_j$ and that by the definition of Z_j it follows that, $Z_j \leq \|f_{\mathscr{P}}(\mathbf{x}) - f_{\mathrm{A}_j}(\mathbf{x})\|$ (where $\mathscr{P} \sim \mathcal{N}_{(\mathrm{A}_j,\sigma^2 I)}$). Then we get,

$$\mathbb{P}\left(Z > \gamma(\mathbf{x})\right) \leq \sum_{j=1}^{k_1} p_j \mathbb{P}_{\mathscr{P} \sim \mathcal{N}_{(\mathrm{A}_j, \sigma^2 I)}} \left(\|f_{\mathscr{P}}(\mathbf{x}) - f_{\mathrm{A}_j}(\mathbf{x})\| > \gamma(\mathbf{x}) \right) \tag{6.23}$$

Now for the kind of nets we consider i.e ones with no bias vectors in any of the layers it follows from the definition of $\{\beta_i\}_{i=1,\ldots,k_1}$ that has been made that the function computed by the net remains invariant if the layer weight $\mathrm{A}_{i,\ell}$ is replaced by $\frac{\beta_i}{\|\mathrm{A}_{i,\ell}\|}\mathrm{A}_{i,\ell}$. And we see that the spectral norm is identically β_i for each layer in this net with modified wights. So we can assume without loss of generality that $\mathrm{A}_{i,\ell} = \beta_i$ for all i and ℓ. Hence it follows that the RHS in equation 6.23 is exactly the quantity for which guarantees have been given in Lemma 6.6.1. And by de-homogenizing the definition of β_i as given in Lemma 6.6.1, the appropriate value of σ can be realized to be the same as given in the theorem statement,

$$\sigma^2 = \frac{1}{2h \log\left(\frac{2dhk_1}{\delta}\right)} \left(\min_{i \in \{1,\ldots,k_1\}} \min \left\{ \frac{\left(\prod_{\ell=1,\ldots,d} \|\mathrm{A}_{i,\ell}\|\right)^{\frac{1}{d}}}{d}, \frac{\gamma(\mathbf{x})}{k_1 edBp_i \left(\prod_{\ell=1,\ldots,d} \|\mathrm{A}_{i,\ell}\|\right)^{1-\frac{1}{d}}} \right\} \right)^2 \tag{6.24}$$

In the above we replace $\gamma(\mathbf{x})$ with $\gamma := \epsilon \max_{\mathbf{x} \in S} \|f_{\mathrm{A}}(\mathbf{x})\|$ and going back to equation 6.18 we can get a concurrent guarantee for all $\mathbf{x} \in S$ as required in the theorem as,

$$\mathbb{P}_{\mathrm{A}' \sim \mathrm{MG(posterior)}} \left(\forall \mathbf{x} \in S, \|f_{\mathrm{A}'}(\mathbf{x}) - f_{\mathrm{A}}(\mathbf{x})\| > 2\gamma \right) \leq \delta$$

□

6.7 Proof of Theorem 6.3.1

Proof. Given that $(\epsilon, \gamma)-$nice w.r.t $\{\mathrm{A}, \mathrm{A}_{i=1,\ldots,k_1}\}$ (as defined in Definition 25), we can invoke Theorem 6.2.1 with $k_1 = 1$ between the trained nets f_{A} and f_{A_i}. We recall the definition of β_i that, $\beta_i^d = \prod_{\ell=1,\ldots,d} \|\mathrm{A}_{i,\ell}\|$ for $\mathrm{A}_{i,\ell}$ being the ℓ^{th} layer matrix corresponding to A_i.

Let the $\beta-$grid be denoted as $\{\tilde{\beta}_k\}$ s.t there exists a point $\tilde{\beta}$ in this grid s.t,

$$|\beta_i - \tilde{\beta}| \leq \frac{\beta_i}{d} \implies \frac{\beta_i^{d-1}}{e} \leq \tilde{\beta}^{d-1} \leq e\beta_i^{d-1} \tag{6.25}$$

Now we recall that by invoking Theorem 6.2.1 on the net f_{A_i}, the σ (in terms of β_i) that would be obtained from there is a maximal choice that the proof could have given us. Hence a smaller value of σ will also give the same guarantees and we go for the following value of the variance defined in terms of the $\tilde{\beta}$ defined above,

$$\tilde{\sigma}^2 := \frac{1}{2h\log\left(\frac{2dh}{\delta}\right)}\left(\min\left\{\frac{\tilde{\beta}}{de^{\frac{1}{d-1}}}, \frac{\frac{\gamma}{8}}{e^2 dB\tilde{\beta}^{d-1}}\right\}\right)^2 \tag{6.26}$$

In the above we have set the p_i parameter of Theorem 6.2.1 to 1 and have also rescaled the γ parameter to $\frac{\gamma}{8}$ so that we have from Theorem 6.2.1 that,

$$\mathbb{P}_{\mathrm{A}'\sim\mathcal{N}(\mathrm{A}_i,\tilde{\sigma}^2 I)}\left[\max_{\mathbf{x}\in S}\|f_{\mathrm{A}'}(\mathbf{x})-f_{\mathrm{A}}(\mathbf{x})\| > 2\times\frac{\gamma}{8}\right] \leq \delta \tag{6.27}$$

$$\tag{6.28}$$

Since, $\max_{\mathbf{x}\in S}\|f_{\mathrm{A}'}(\mathbf{x})-f_{\mathrm{A}}(\mathbf{x})\| < \frac{\gamma}{4} \implies \max_{\mathbf{x}\in S}\|f_{A'}(\mathbf{x})-f_{A}(\mathbf{x})\|_\infty < \frac{\gamma}{4}$ we have,

$$\mathbb{P}_{\mathrm{A}'\sim\mathcal{N}(\mathrm{A}_i,\tilde{\sigma}^2 I)}\left[\max_{\mathbf{x}\in S}\|f_{\mathrm{A}'}(\mathbf{x})-f_{\mathrm{A}}(\mathbf{x})\|_\infty < \frac{\gamma}{4}\right] \geq 1-\delta$$

Hence we are in the situation whereby Theorem 6.1.2 can be invoked with $\mathrm{A} = \mathbf{w}$ and $\mu_{\mathbf{w}} = \mathcal{N}(\mathrm{A}_i, \tilde{\sigma}^2 I)$ to guarantee that there exists a distribution $\tilde{\mu}_{\mathbf{w}}$ s.t the following inequality holds with probability at least $1-\delta$ over sampling m sized data-sets

$$\mathbb{E}_{\mathrm{A}+\tilde{\mathbf{u}}\sim\tilde{\mu}_{\mathrm{A}}}[L_0(f_{\mathrm{A}+\tilde{\mathbf{u}}})] \leq \hat{L}_{\frac{\gamma}{2}}(f_{\mathrm{A}}) + \sqrt{\frac{\mathrm{KL}(\mathcal{N}(\mathrm{A}_i,\tilde{\sigma}^2 I)||P) + \log\frac{3m}{\delta}}{m-1}} \tag{6.29}$$

Because the grid of $\{\tilde{\beta}_k\}$ was pre-fixed we can choose the prior distribution P in a data-dependent way from the grid of priors specified in the Definition 26 with their variance being $\tilde{\sigma}$ as given in Definition 6.26, determined by a chosen element from this set $\{\tilde{\beta}_k\}$! Now we recall the definition of the net weights, $\{\mathrm{B}_{\lambda^*,j}\}_{j=1,\ldots,k_1}$ in the theorem statement and we choose,

$$P := \text{ a distribution s.t its p.d.f is } \frac{1}{k_1}\sum_{j=1}^{k_1}\mathcal{N}_{(\mathrm{B}_{\lambda^*,j},\tilde{\sigma}^2 I)}$$

And this means that for the KL-term above we can invoke Theorem 6.A.1 with $f = \mathcal{N}(\mathrm{A}_i, \tilde{\sigma}^2 I)$ and $\mathrm{GM} = P$ to get,

$$\mathrm{KL}(\mathcal{N}(\mathrm{A}_i, \tilde{\sigma}^2 I)||P) \leq -\log\left[\frac{1}{k_1}\sum_{j=1}^{k_1} e^{-\frac{1}{2\tilde{\sigma}^2}\|\mathrm{A}_i - \mathrm{B}_{\lambda^*,j}\|^2}\right] \tag{6.30}$$

We recall that our angle grid (which determines the value of λ^* above as described in the theorem statement) was of size 314 and K_1 was the size of the $\beta-$grid and thus the size of the full grid of priors is $314K_1 =: K$

Hence its clear as to how equation 6.29 holding true for each of the K possible choices of P decided by the mechanism described above, satisfies the required hypothesis for Theorem 6.B.1 to be invoked. The required theorem now follows by further upperbounding the KL-term in equation 6.29 as given in equation 6.30.

Now we are left with having to specify the required grid $\{\tilde{\beta}_k\}_{k=1,\ldots,K_1}$ so that we are always guaranteed to find a $\tilde{\beta}$ as given in Definition 6.25. Towards that we upperbound equation 6.30 as follows,

$$\mathrm{KL}(\mathcal{N}(\mathrm{A}_i, \tilde{\sigma}^2 I)||P) \leq -\log\left[\frac{1}{k_1}\sum_{j=1}^{k_1} e^{-\frac{1}{2\tilde{\sigma}^2}\|\mathrm{A}_i - \mathrm{B}_{\lambda^*,j}\|^2}\right] \leq -\log\left[\frac{1}{k_1}\sum_{j=1}^{k_1} e^{-\frac{1}{2\sigma^2}\|\mathrm{A}_i - \mathrm{B}_{\lambda^*,j}\|^2}\right] \tag{6.31}$$

where we have gotten the second inequality by recalling Definitions 6.26, 6.25 and defining,

$$\sigma^2 := \frac{1}{2h\log(2dh/\delta)}\left(\min\left\{\frac{\beta_i}{de^{\frac{2}{d-1}}}, \frac{\gamma/8}{e^3 dB\beta_i^{d-1}}\right\}\right)^2 = \left(\frac{\beta_i \exp -\frac{2}{(d-1)}}{d\sqrt{2h\log\left(\frac{2dh}{\delta}\right)}}\right)^2 \min\left\{\frac{\gamma^2}{(8B\beta_i^d \exp(3-\frac{2}{d-1}))^2}, 1\right\}$$

Now we observe the following,

1. When $\beta_i \leq \left(\frac{\gamma}{2B}\right)^{1/d}$ this implies $\|f_{\mathrm{A}_i}(\mathbf{x})\| \leq \frac{\gamma}{2}$ which implies $\hat{L}_\gamma = 1$ by definition. Therefore equation 6.29 holds trivially.

2. We can ask when is it that the upperbound on the KL term given in equation 6.31 in terms of this σ such that the resultant upperbound on $\sqrt{\frac{2\mathrm{KL}(\mathcal{N}(\mathrm{A}_i,\tilde{\sigma}^2 I)||P)}{m-1}}$ (when substituted into equation 6.29) greater than 1 i.e the range of β_i for which the following inequality holds (and thus making the inequality 6.29 hold trivially by ensuring that the ensuing upperbound on the RHS of it is greater than 1),

$$1 \leq \frac{-\log\left[\frac{1}{k_1}\sum_{j=1}^{k_1} e^{-\frac{1}{2\sigma^2}\|\mathrm{A}_i-\mathrm{B}_{\lambda^*,j}\|^2}\right]}{m-1} = \frac{1}{m-1}\log\left[\frac{1}{\frac{1}{k_1}\sum_{j=1}^{k_1} e^{-\frac{1}{2\sigma^2}\|\mathrm{A}_i-\mathrm{B}_{\lambda^*,j}\|^2}}\right] \tag{6.32}$$

$$\begin{aligned}\log\left[\frac{1}{\frac{1}{k_1}\sum_{j=1}^{k_1} e^{-\frac{1}{2\sigma^2}\|\mathrm{A}_i-\mathrm{B}_{\lambda^*,j}\|^2}}\right] &\geq \log\left[\frac{1}{\frac{1}{k_1}\sum_{j=1}^{k_1} \max_{i,j}\{e^{-\frac{1}{2\sigma^2}\|\mathrm{A}_i-\mathrm{B}_{\lambda^*,j}\|^2}\}}\right] \\ &= \log\left[\frac{1}{\frac{1}{k_1}\sum_{j=1}^{k_1} e^{-\frac{1}{2\sigma^2}\min_{i,j}\{\|\mathrm{A}_i-\mathrm{B}_{\lambda^*,j}\|^2\}}}\right]\end{aligned}$$

Now we invoke the definition of $d_{\min}$ to get,

$$\log\left[\frac{1}{\frac{1}{k_1}\sum_{j=1}^{k_1} e^{-\frac{1}{2\sigma^2}\min\{\|\mathrm{A}_i-\mathrm{B}_{\lambda^*,j}\|^2\}}}\right] \geq \frac{d_{\min}^2}{2\sigma^2}$$

Thus substituting back into equation 6.32 we see that a sufficient condition for 6.32 to be satisfied is,

$$\left(\frac{\beta_i \exp -\frac{2}{(d-1)}}{d\sqrt{2h\log\left(\frac{2dh}{\delta}\right)}}\right)^2 \min\left\{\frac{\gamma^2}{(8B\beta_i^d \exp\left(3-\frac{2}{d-1}\right))^2}, 1\right\} = \sigma^2 \leq \frac{d_{\min}^2}{2(m-1)}$$

A further sufficient condition for the above to be satisfied is,

$$\left(\frac{\beta_i \exp -\frac{2}{(d-1)}}{d\sqrt{2h\log\left(\frac{2dh}{\delta}\right)}}\right)^2 \cdot \frac{\gamma^2}{(8B\beta_i^d \exp\left(3-\frac{2}{d-1}\right))^2} \leq \frac{d_{\min}^2}{2(m-1)}$$

The above leads to the constraint, $\beta_i \geq \left(\frac{\gamma\sqrt{2(m-1)}}{(8Be^3 d d_{\min})\sqrt{2h\log\left(\frac{2dh}{\delta}\right)}}\right)^{\frac{1}{(d-1)}}$

Thus we combine the two points above to see that a relevant interval of β_i is,

$$\left[\left(\frac{\gamma}{2B}\right)^{\frac{1}{d}}, \left(\frac{\gamma\sqrt{2(m-1)}}{(8Be^3 d d_{\min})\sqrt{2h\log\left(\frac{2dh}{\delta}\right)}}\right)^{\frac{1}{(d-1)}}\right]$$

We recall that the parameters have been chosen so that the above interval is non-empty.

We note that if we want a grid on the interval $[a, b]$ s.t for every value $x \in [a, b]$ there is a grid-point g s.t $|x - g| \leq \frac{x}{d}$ then a grid size of $\frac{bd}{2a}$ suffices. [4] Hence the a grid of the following size K_1 suffices for us to capture with needed accuracy all the possible values of β_i,

$$K_1 = \frac{d}{2} \times \frac{\left(\frac{\gamma\sqrt{2(m-1)}}{(8Be^3 d d_{\min})\sqrt{2h\log\left(\frac{2dh}{\delta}\right)}}\right)^{\frac{1}{(d-1)}}}{\left(\frac{\gamma}{2B}\right)^{\frac{1}{d}}}$$

And thus we have specified the "beta-grid" as mentioned in Definition 26. □

6.8 Conclusion

We conclude by reporting two other observations that have come to light from the experiments above. *Firstly,* We have also done experiments (not reported here) where we have used the CIFAR data as a binary classification task and there we observed that the angular deflection under training is significantly lower than whats reported above for CIFAR−10. In such situations when this deflection is lower the relative advantage of our bound over Neyshabur et al., 2017's bound is even greater. *Secondly,* We have additionally also observed that the maximum angle between A and any of the A_is is typically $40 - 60\%$ larger than the corresponding angular spread of the intial cluster i.e the maximum angle between B and any of the B_is. Note that all the final cluster nets are approximately of the same accuracy. Thus nets initialized close by to each other often seem to not end up as close post-training even when trained to the same accuracy on the same data and using the same algorithm. We believe that this dispersion behaviour of nets warrants further investigation.

[4] If g is the grid point which is the required approximation to x i.e $|x - g| \leq \frac{x}{d} \implies x \in \left(\frac{d}{d+1}g, \frac{d}{d-1}g\right)$ Since $a \leq g \implies \frac{2da}{d^2-1} \leq \left(\frac{d}{d-1} - \frac{d}{d+1}\right)\tilde{\beta}$. So $\frac{2da}{(d^2-1)}$ is the smallest grid spacing that might be needed and hence the maximum number number of grid points needed is $\frac{(b-a)(d^2-1)}{2ad} < \frac{(b-a)d}{2a} < \frac{bd}{2a}$

Given the demonstrated advantages of our PAC-Bayesian bounds on neural nets we believe that these observations deserve further investigation and being able to theoretically explain them and incorporate them into the PAC-Bayesian framework might contribute towards getting even better bounds.

Appendix To Chapter 6

6.A The KL upperbounds

The normal distributions $\mathcal{N}_{\mathrm{A},\sigma^2}$ as stated in definition 23 can be made explicit as,

$$\log \mathcal{N}_{(\mathrm{A},\sigma^2)}(\mathbf{w}) = -\frac{1}{2}\left[\frac{\|\mathbf{w}-\mathrm{A}\|^2}{\sigma^2} + \dim(\mathrm{A})\log\left(2\pi\sigma^2\right)\right] \tag{6.33}$$

Theorem 6.A.1. Assume being given distributions f and GM on $\mathbb{R}^n$. f is Gaussian and has mean $\bar{} = \mathrm{A}$ and covariance $\Sigma_f = \sigma_f^2 I_n$ for some $\sigma_f > 0$. While GM is a Gaussian mixture with k_1 Gaussians $\{f_{\mathrm{GM},r}\}_{r=1,\ldots,k_1}$ with weights, $\{a_{\mathrm{GM},i} \geq 0\}_{i=1,..,k_1}$ and $\sum_{i=1}^{k_1} a_{\mathrm{GM},i} = 1$ s.t the k_1 components have means as $\{\bar{}_{\mathrm{GM},r} = \mathrm{B}_r\}_{r=1,\ldots,k_1}$ and the covariance matrix of the components as $\Sigma_{\mathrm{GM}} := \sigma_{\mathrm{GM}}^2 I_n$ for some $\sigma_2 > 0$. Then we have the following upperbound,

$$\mathrm{KL}(f\|GM) \leq \frac{n}{2}\left(\frac{\sigma_f^2}{\sigma_{\mathrm{GM}}^2} - 1\right) + n\log(\frac{\sigma_{\mathrm{GM}}}{\sigma_f}) - \log\left[\sum_{r=1}^{k_1} a_{\mathrm{GM},r} e^{-\frac{1}{2\sigma_{\mathrm{GM}}^2}\|\mathrm{A}-\mathrm{B}_r\|^2}\right]$$

Proof. We recall the following theorem,

Theorem 6.A.2 (Durrieu-Thiran-Kelly (ICASSP 2012))**.** Given the definitions of f and GM as above but with Σ_f and Σ_{GM} being generic we have as upperbound for the KL divergence between the two distributions,

$$\mathrm{KL}(f\|\mathrm{GM}) \leq \left(H(f) + \log\frac{\exp(L_f(f))}{\sum_{r=1}^{k_2} a_{\mathrm{GM},r} e^{-D_{KL}(f\|f_{\mathrm{GM},r})}}\right),$$

where $H(f) := \mathbb{E}\left[-\log(f(x))\right] =: -L_f(f)$ □

Now using the definition of $\mu_{\mathrm{GM},r}$ and Σ_{GM} and known expression for the KL divergence between 2 Gaussians the upperbound given above simplifies as,

$$\mathrm{KL}(f\|\mathrm{GM}) \leq -\log\left[\sum_{r=1}^{k_1} a_{\mathrm{GM},r} e^{-\mathrm{KL}(f\|f_{\mathrm{GM},r})}\right] = -\log\left[\sum_{r=1}^{k_1} a_{\mathrm{GM},r}\sqrt{\frac{e^n \det(\Sigma_f)}{\det(\Sigma_{\mathrm{GM}})\det(e^{\Sigma_{\mathrm{GM}}^{-1}\Sigma_f})}} e^{-\frac{1}{2}(\bar{-}-\bar{-}_{\mathrm{GM},r})^T\Sigma_{\mathrm{GM}}^{-1}(\bar{-}-\bar{-}_{\mathrm{GM},r})}\right]$$

Now we recall the definitions of A, $\{\mathrm{B}_r\}_{r=1,\ldots,k_1}, \sigma_f$ and σ_{GM} to further simplify the above to get,

$$\begin{aligned}
\mathrm{KL}(f\|\mathrm{GM}) &\leq -\log\sqrt{\frac{e^n\det(\sigma_f^2 I)}{\det(\sigma_{\mathrm{GM}}^2 I)\det\left(e^{(\sigma_{\mathrm{GM}}^2 I)^{-1}(\sigma_f^2 I)}\right)}} - \log\left[\sum_{r=1}^{k_2} a_{\mathrm{GM},r} e^{-\frac{1}{2}(\mathrm{A}-\mathrm{B}_r)^T(\sigma_{\mathrm{GM}}^2 I)^{-1}(\mathrm{A}-\mathrm{B}_r)}\right] \\
&= \log\sqrt{\frac{\det(\sigma_{\mathrm{GM}}^2 I)\det\left(e^{(\sigma_{\mathrm{GM}}^2 I)^{-1}(\sigma_f^2 I)}\right)}{e^n\det(\sigma_f^2 I)}} - \log\left[\sum_{r=1}^{k_1} a_{\mathrm{GM},r} e^{-\frac{1}{2\sigma_{\mathrm{GM}}^2}\|\mathrm{A}-\mathrm{B}_r\|^2}\right] \\
&\leq \frac{n}{2}\left(\frac{\sigma_f^2}{\sigma_{\mathrm{GM}}^2} - 1\right) + n\log\left(\frac{\sigma_{\mathrm{GM}}}{\sigma_f}\right) - \log\left[\sum_{r=1}^{k_1} a_{\mathrm{GM},r} e^{-\frac{1}{2\sigma_{\mathrm{GM}}^2}\|\mathrm{A}-\mathrm{B}_r\|^2}\right]
\end{aligned}$$

And thus the required theorem is proven. □

6.B Data-Dependent Priors

We continue in the same notation as used in Theorem 6.1.1 and we prove the following theorem.

Theorem 6.B.1. Suppose we have K prior distributions $\{\pi_i\}_{i=1,\ldots,K}$ s.t for some $\delta > 0$ the following inequality holds for each π_i

$$\mathbb{P}_{S\sim\mathcal{D}^m(\mathrm{Z})}\left[\forall Q\, \mathbb{E}_{h\sim Q}[L(h)] \leq \mathbb{E}_{h\sim Q}[\hat{L}(h)] + \sqrt{\frac{\mathrm{KL}(Q\|\pi_i) + \log\frac{m}{\delta}}{m-1}}\right] \geq 1-\delta. \tag{6.34}$$

Then,

$$\mathbb{P}_{S\sim\mathcal{D}^m(\mathrm{Z})}\left[\forall i \in \{1,\ldots,K\}\,\forall Q\, \mathbb{E}_{h\sim Q}[L(h)] \leq \mathbb{E}_{h\sim Q}[\hat{L}(h)] + \sqrt{\frac{\mathrm{KL}(Q\|\pi_i) + \log\frac{m}{\delta}}{m-1}}\right] \geq 1-K\delta. \tag{6.35}$$

Proof. Let f be a real-valued function on the space of distributions and data sets. Let $E_Q = \{S \mid f(S,Q) > 0\}$. And note that $\cap_Q E_Q = \{S \mid \forall Q f(S,Q) > 0\}$. Therefore $\mathbb{P}_S[S \mid \forall Q f(S,Q) > 0] = \mathbb{P}_S[\cap_Q E_Q]$ regardless of the distribution on S.

Consider K functions $f_1,\ldots,f_K$ s.t for some $\delta \in [0,1]$ we have,

$$\forall i \in \{1,\dots,K\}\ \mathbb{P}_S[S \mid \forall Q\ f_i(S,Q) > 0] \geq 1-\delta$$

Now we define the events $E_{i,Q} = \{S \mid f_i(S,Q) > 0\}$ for each $i \in \{1,\dots,K\}$. Thus we can deduce the following:

$$\begin{aligned}
&\forall i \in \{1,\dots,K\}\ \mathbb{P}_S[S \mid \forall Q\ f_i(S,Q) > 0] && \geq 1-\delta \\
\implies &\forall i \in \{1,\dots,K\}\ \mathbb{P}_S[\cap_Q E_{i,Q}] && \geq 1-\delta \quad \text{(as discussed in the beginning of the proof)} \\
\implies &\forall i \in \{1,\dots,K\}\ \mathbb{P}_S[\cup_Q E^c_{i,Q}] && \leq \delta \\
\implies &\mathbb{P}_S[\cup_{i=1}^K \Big(\cup_Q E^C_{i,Q} \Big)] && \leq K\delta \\
\implies &\mathbb{P}_S[\cap_{i=1}^K \cap_Q E_{i,Q}] && \geq 1-K\delta \\
\implies &\mathbb{P}_S[S \mid \forall Q\ \forall i \in \{1,\dots,K\}\ f_i(S,Q) > 0] && \geq 1-K\delta
\end{aligned} \tag{6.36}$$

Now we use $f_i(S,Q) = \mathbb{E}_{h\sim Q}[\hat{L}(h)] - \mathbb{E}_{h\sim Q}[L(h)] + \sqrt{\frac{\mathrm{KL}(Q\|\pi_i)+\log\frac{m}{\delta}}{m-1}}$ in equation (6.36) and use condition (6.34) to get result (6.35). □

The above theorem encapsulates what is conventionally called as using a "data-dependent prior". This is because if we can construct a list of K priors and work with $\delta < \frac{1}{K}$ then the above theorem lets us choose in a data (i.e the training data S) dependent way not just the posterior distribution Q but also the prior π_i from the list and still assures us a high probability upperbound on the difference between the true risk and the empirical risk of the stochastic classifier.

6.C Proof of Theorem 6.1.2

Proof. Given a predictor weight $\mathbf{w}$, for explicitness we will denote by $\mu_{\mathbf{w}}$ and $\tilde{\mu}_{\mathbf{w}}$ what were called μ and $\tilde{\mu}$ in the theorem statement.

We will first isolate its set of "good" perturbations i.e we define the following set $S_{\mathbf{w}}$ as,

$$S_{\mathbf{w}} = \left\{ \mathbf{w}+\mathbf{u} \mid \max_{\mathbf{x}\in S} \|f_{\mathbf{w}+\mathbf{u}}(\mathbf{x}) - f_{\mathbf{w}}(\mathbf{x})\|_\infty < \frac{\gamma}{4} \right\}$$

Corresponding to the predictor weight $\mathbf{w}$, let $\mu'_{\mathbf{w}}$ be a distribution on the weights s.t the required condition holds i.e,

$$\mathbb{P}_{\mathbf{u}\sim\mu'_{\mathbf{w}}}\left[\max_{\mathbf{x}\in S}\|f_{\mathbf{w}+\mathbf{u}}(\mathbf{x})-f_{\mathbf{w}}(\mathbf{x})\|_\infty<\frac{\gamma}{4}\right]\geq\frac{1}{2}$$

Let $\mathbf{w}+\mathbf{u}\sim\mu_{\mathbf{w}}$ when $\mathbf{u}\sim\mu'_{\mathbf{w}}$.

Now we define the quantity, $Z(\mu_{\mathbf{w}}):=\mathbb{P}_{\mathbf{w}+\mathbf{u}\sim\mu_{\mathbf{w}}}[\mathbf{w}+\mathbf{u}\in S_{\mathbf{w}}]$. From the definition it follows that, $Z(\mu_{\mathbf{w}})\geq\frac{1}{2}$. Now let us define another distribution $\tilde{\mu}_{\mathbf{w}}$ over the set of predictor's weights s.t the p.d.fs are related as follows (in the following we overload the notation of the distribution to also denote the corresponding p.d.f),

$$\tilde{\mu}_{\mathbf{w}}(x)=\frac{\mu_{\mathbf{w}}(x)}{Z(\mu_{\mathbf{w}})}\delta_{x\in S_{\mathbf{w}}}$$

Thus $\tilde{\mu}_{\mathbf{w}}$ is supported on $S_{\mathbf{w}}$. From the above definition it follows that if $\mathbf{w}+\tilde{\mathbf{u}}\sim\tilde{\mu}_{\mathbf{w}}$ then $\max_{\mathbf{x}\in S}\|f_{\mathbf{w}+\tilde{\mathbf{u}}}(\mathbf{x})-f_{\mathbf{w}}(\mathbf{x})\|_\infty<\frac{\gamma}{4}$ which is equivalent to $\max_{i\in\{1,2,..,k\},\mathbf{x}\in S}\|f_{\mathbf{w}+\tilde{\mathbf{u}}}(\mathbf{x})[i]-f_{\mathbf{w}}(\mathbf{x})[i]\|<\frac{\gamma}{4}$. This in turn implies,

$$\max_{i,j\in\{1,2,..,k\},\mathbf{x}\in S}\|(f_{\mathbf{w}+\tilde{\mathbf{u}}}(\mathbf{x})[i]-f_{\mathbf{w}+\tilde{\mathbf{u}}}(\mathbf{x})[j])-(f_{\mathbf{w}}(\mathbf{x})[i]-f_{\mathbf{w}}(\mathbf{x})[j])\|<\frac{\gamma}{2}\tag{6.37}$$

Which in turn implies the following inequality, [5]

$$\hat{L}_0(f_{\mathbf{w}+\tilde{\mathbf{u}}})\leq\hat{L}_{\gamma/2}(f_{\mathbf{w}})\tag{6.38}$$

Similar to $\tilde{\mu}$ we define the following,

$$\tilde{\mu}^c_{\mathbf{w}}(\mathbf{x})=\frac{1}{1-Z(\mu_{\mathbf{w}})}\mu_{\mathbf{w}}(\mathbf{x})\delta_{\mathbf{x}\in S^c_w}$$

[5] We give the proof in this footnote for convenience. Let j be arbitrary, let $i=\arg\max_{i\neq j}f_{\mathbf{w}+\tilde{\mathbf{u}}}(\mathbf{x})[i]$. Let $S_j=\{x\mid f_{\mathbf{w}}(\mathbf{x})[i]-f_{\mathbf{w}}(\mathbf{x})[j]\geq-\gamma/2\}$, and $S'_j=\{x\mid f_{\mathbf{w}+\tilde{\mathbf{u}}}(\mathbf{x})[i]-f_{\mathbf{w}+\tilde{\mathbf{u}}}(\mathbf{x})[j]\geq 0\}$. Clearly $S'_j\subseteq S_j$ because $f_{\mathbf{w}+\tilde{\mathbf{u}}}(\mathbf{x})[i]-f_{\mathbf{w}+\tilde{\mathbf{u}}}(\mathbf{x})[j]\geq 0\implies f_{\mathbf{w}}(\mathbf{x})[i]-f_{\mathbf{w}}(\mathbf{x})[j]\geq-\gamma/2$ from (6.37).

Now recall that we can write, $L_{\gamma/2}(f_{\mathbf{w}})=\mathbb{E}_y[\mathbb{E}_{\mathbf{x}}[\mathbf{1}(f_{\mathbf{w}}(\mathbf{x})[y]\leq\gamma/2+\max_{i\neq y}f_{\mathbf{w}}(\mathbf{x})[i])\mid y]$ and $L_0(f_{\mathbf{w}+\tilde{\mathbf{u}}})=\mathbb{E}_y[\mathbb{E}_{\mathbf{x}}[\mathbf{1}(f_{\mathbf{w}+\tilde{\mathbf{u}}}(\mathbf{x})[y]\leq\max_{i\neq y}f_{\mathbf{w}+\tilde{\mathbf{u}}}(\mathbf{x})[i])\mid y]$ Hence equation (6.38) follows by noting that the support of the indicator function in $L_0(f_{\mathbf{w}+\tilde{\mathbf{u}}})$ is contained in the support of the indicator function in $L_{\gamma/2}(f_{\mathbf{w}})$ because $S'_j\subseteq S_j$.

Thus substituting the expressions for $\tilde{\mu}_{\mathbf{w}}$ and $\tilde{\mu}^c_{\mathbf{w}}$ for the given prior P we have,

$$\begin{aligned}
&\mathrm{KL}(\mu_{\mathbf{w}} \mid P) \\
&= \int \mu_{\mathbf{w}} \log\left(\frac{P}{\mu_{\mathbf{w}}}\right) = \int_{S_w} \mu_{\mathbf{w}} \log\left(\frac{P}{\mu_{\mathbf{w}}}\right) + \int_{S_w^c} \mu_{\mathbf{w}} \log\left(\frac{P}{\mu_{\mathbf{w}}}\right) \\
&= \int Z(\mu_{\mathbf{w}})\tilde{\mu}_{\mathbf{w}} \log\left(\frac{P}{Z(\mu_{\mathbf{w}})\tilde{\mu}_{\mathbf{w}}}\right) + \int (1 - Z(\mu_{\mathbf{w}}))\tilde{\mu}^c_{\mathbf{w}} \log\left(\frac{P}{(1 - Z(\mu_{\mathbf{w}}))\tilde{\mu}^c_{\mathbf{w}}}\right) \\
&= Z(\mu_{\mathbf{w}}) \left[-\log Z(\mu_{\mathbf{w}}) + \int \tilde{\mu}_{\mathbf{w}} \log\left(\frac{P}{\tilde{\mu}_{\mathbf{w}}}\right)\right] \\
&\quad + (1 - Z(\mu_{\mathbf{w}})) \left[-\log(1 - Z(\mu_{\mathbf{w}})) + \int \tilde{\mu}^c_{\mathbf{w}} \log\left(\frac{P}{\tilde{\mu}^c_{\mathbf{w}}}\right)\right] \\
&= Z(\mu_{\mathbf{w}})\mathrm{KL}(\tilde{\mu}_{\mathbf{w}} \mid P) + (1 - Z(\mu_{\mathbf{w}}))\mathrm{KL}(\tilde{\mu}^c_{\mathbf{w}} \mid P) \\
&\quad - Z(\mu_{\mathbf{w}}) \log Z(\mu_{\mathbf{w}}) - (1 - Z(\mu_{\mathbf{w}})) \log(1 - Z(\mu_{\mathbf{w}})) \\
\implies \mathrm{KL}(\tilde{\mu}_{\mathbf{w}} \mid P) &= \frac{1}{Z(\mu_{\mathbf{w}})} \Big\{ \mathrm{KL}(\mu_{\mathbf{w}} \mid P) - (1 - Z(\mu_{\mathbf{w}}))\mathrm{KL}(\tilde{\mu}^c_{\mathbf{w}} \mid P) \\
&\qquad + Z(\mu_{\mathbf{w}}) \log Z(\mu_{\mathbf{w}}) + (1 - Z(\mu_{\mathbf{w}})) \log(1 - Z(\mu_{\mathbf{w}})) \Big\}
\end{aligned} \tag{6.39}$$

We recall that for any $Z \in [\frac{1}{2}, 1]$ we have, $|Z \log Z + (1 - Z) \log(1 - Z)| \leq 1$ and $\mathrm{KL}(\tilde{\mu}^c_{\mathbf{w}} \| P) \geq 0$. Thus we have,

$$\mathrm{KL}(\tilde{\mu}_{\mathbf{w}} \| P) \leq \frac{1}{Z(\mu_{\mathbf{w}})} \left(\mathrm{KL}(\mu_{\mathbf{w}} \| P) + 1\right) \leq 2 \left(\mathrm{KL}(\mu_{\mathbf{w}} \| P) + 1\right) \tag{6.40}$$

We remember that above we had sampled the weights of the perturbed net $f_{\mathbf{w}+\tilde{\mathbf{u}}}$ as $\mathbf{w} + \tilde{\mathbf{u}} \sim \tilde{\mu}_{\mathbf{w}}$. Now we write the highly likely event guaranteed by PAC-Bayesian bounds in Theorem 6.1.1, for the margin loss $L_{\frac{\gamma}{2}}$ evaluated on the predictor $f_{\mathbf{w}}$ for some prior P and a (data dependent) choice of posterior $\tilde{\mu}_{\mathbf{w}}$. Further we invoke equation (6.38) and (6.40) on that, to get the following,

$$\begin{aligned}
\mathbb{E}_{\mathbf{w}+\tilde{\mathbf{u}}\sim\tilde{\mu}_{\mathbf{w}}}[L_0(f_{\mathbf{w}+\tilde{\mathbf{u}}})] &\leq \mathbb{E}_{\mathbf{w}+\tilde{\mathbf{u}}\sim\tilde{\mu}_{\mathbf{w}}}[\hat{L}_0(f_{\mathbf{w}+\tilde{\mathbf{u}}})] + \sqrt{\frac{\mathrm{KL}(\tilde{\mu}_{\mathbf{w}}||P) + \log\frac{m}{\delta}}{2(m-1)}} \\
&\leq \hat{L}_{\gamma/2}(f_{\mathbf{w}}) + \sqrt{\frac{\mathrm{KL}(\tilde{\mu}_{\mathbf{w}}||P) + \log\frac{m}{\delta}}{2(m-1)}} \\
&\leq \hat{L}_{\gamma/2}(f_{\mathbf{w}}) + \sqrt{\frac{2\mathrm{KL}(\mu_{\mathbf{w}}||P) + 2 + \log\frac{m}{\delta}}{2(m-1)}} = \hat{L}_{\gamma/2}(f_{\mathbf{w}}) + \sqrt{\frac{\mathrm{KL}(\mu_{\mathbf{w}}||P) + 1 + \frac{1}{2}\log\frac{m}{\delta}}{m-1}} \\
&\leq \hat{L}_{\gamma/2}(f_{\mathbf{w}}) + \sqrt{\frac{\mathrm{KL}(\mu_{\mathbf{w}}||P) + \log\frac{3m}{\delta}}{m-1}}
\end{aligned}$$

where in the last line we have used, $1 + \frac{1}{2}\log\frac{m}{\delta} < 1 + \log\frac{m}{\delta} < \log 3 + \log\frac{m}{\delta}$

□

6.D Proof of Theorem 6.1.3

Proof. Firstly we observe the following Theorem 6.D.1 which is a slight variation of a lemma in Neyshabur et al., 2017. The proof of Theorem 6.D.1 follows from exactly the same arguments as was needed to prove Theorem 6.1.2.

Theorem 6.D.1. Let $f_{\mathbf{w}} : \chi \to \mathbb{R}^k$ be any predictor with parameters $\mathbf{w}$ and lets use the margin loss as defined in equation 22. Let P be any distribution (the "data-independent prior") on the space of parameters of f and $\mathcal{D}$ be a distribution on χ. Let it be true that for some $\gamma > 0$, we know of distributions $\mu'_{\mathbf{w}}$ and $\mu_{\mathbf{w}}$ (depending on the weight $\mathbf{w}$ of the given predictor) on the space of parameters of the predictor s.t,

$$\mathbb{P}_{\mathbf{u}\sim\mu'_{\mathbf{w}}}\left[\sup_{\mathbf{x}\in\chi}\|f_{\mathbf{w}+\mathbf{u}}(\mathbf{x}) - f_{\mathbf{w}}(\mathbf{x})\|_\infty < \frac{\gamma}{4}\right] \geq \frac{1}{2} \text{ and } \mathbf{w}+\mathbf{u} \sim \mu_{\mathbf{w}} \tag{6.41}$$

Then for any $\delta \in [0,1]$ the following guarantee holds,

$$\begin{aligned}
\mathbb{P}_{\chi\sim\mathcal{D}^m(\chi)}\Bigg[&\forall \mathbf{w} \text{ and corresponding } \mu_{\mathbf{w}} \text{ s.t condition 6.41 holds}, \exists\, \tilde{\mu}_{\mathbf{w}} \text{ s.t} \\
&\mathbb{E}_{\mathbf{w}+\tilde{\mathbf{u}}\sim\tilde{\mu}_{\mathbf{w}}}[L_0(f_{\mathbf{w}+\tilde{\mathbf{u}}})] \leq \hat{L}_{\frac{\gamma}{2}}(f_{\mathbf{w}}) + \sqrt{\frac{\mathrm{KL}(\mu_{\mathbf{w}}||P) + \log\frac{3m}{\delta}}{m-1}}\Bigg] \geq 1-\delta
\end{aligned} \tag{6.42}$$

□

The following Theorem 6.D.2 from Neyshabur et al., 2017 is a bound for neural net functions under controlled perturbations and we state it without proof.

Theorem 6.D.2 (Neural net perturbation bound of Neyshabur et al., 2017). Let us be given a depth d neural net, $f_{\mathbf{w}}$, with width h and weight vector $\mathbf{w}$ which is mapping, $B_n(B) \to \mathbb{R}^k$ where $B_n(B)$ is the radius B ball around the origin in $\mathbb{R}^n$. Now cosider a perturbation on the weights given by, $\mathbf{u} = \text{vec}(\{U_\ell\}_{\ell=1}^d)$ s.t $\|U_\ell\|_2 \leq \frac{1}{d}\|W_\ell\|_2$. Then we have for all $\mathbf{x} \in B_n(B)$,

$$\|f_{\mathbf{w}+\mathbf{u}}(\mathbf{x}) - f_{\mathbf{w}}(\mathbf{x})\|_2 \leq eB(\prod_{\ell=1}^{d}\|W_\ell\|_2)\sum_{\ell=1}^{d}\frac{\|U_\ell\|_2}{\|W_\ell\|_2} \tag{6.43}$$

□

Now, for some $\sigma > 0$ consider the random variable $\mathbf{u} \sim \mathcal{N}(0, \sigma^2 I)$ where $\mathbf{u}$ is imagined as the vector of the weights of the neural net f in the above theorem. Let $\{U_\ell \in \mathbb{R}^{h\times h}\}_{\ell=1}^d$ be the matrices of the neural net corresponding to $\mathbf{u}$.

We can define matrices $\{\mathrm{B}_p \in \mathbb{R}^{h\times h} \mid p = 1, \ldots, h^2\}$ s.t each B_p has σ in an unique entry of it and all other entries are 0. Then it follows that as random matrices, $U_\ell = \sum_{p=1}^{h^2}\gamma_p \mathrm{B}_p$ with $\gamma_p \sim \mathcal{N}(0,1)$. We note that $\|\sum_{p=1}^{h^2}\mathrm{B}_p\mathrm{B}_p^\top\| = h\cdot\sigma^2$ since h is the largest eigenvalue of an all ones $h-$dimensional square matrix. Now we invoke Corollary 4.2 of Tropp, 2012 here to get for any $t > 0$,

$$\mathbb{P}_{U_\ell}\left[\|U_{i,\ell}\|_2 > t\right] \leq 2he^{-\frac{t^2}{2h\sigma^2}} \tag{6.44}$$

Using union bound for the d layer matrices of $\mathbf{u}$ we get,

$$\mathbb{P}_{\{U_\ell\sim\mathcal{N}(0,\sigma^2 I_{h\times h})\}_{\ell=1,\ldots,d}}[\exists i \text{ s.t } \|U_\ell\| > t] \leq 2dhe^{-\frac{t^2}{2h\sigma^2}}$$

This is equivalent to,

$$\mathbb{P}_{\{U_\ell\sim\mathcal{N}(0,\sigma^2 I_{h\times h})\}_{\ell=1,\ldots,d}}[\forall i\|U_\ell\| \leq t] \geq 1 - 2dhe^{-\frac{t^2}{2h\sigma^2}} \tag{6.45}$$

If $t = \sigma\sqrt{2h\log(4dh)}$ then $1 - 2dhe^{-\frac{t^2}{2h\sigma^2}} = \frac{1}{2}$ and so we have,

$$\mathbb{P}_{\{U_\ell \sim \mathcal{N}(0,\sigma^2 I_{h\times h})\}_{\ell=1,\ldots,d}}[\forall i \|U_\ell\| \leq \sigma\sqrt{2h\log(4dh)}] \geq \frac{1}{2} \tag{6.46}$$

Now corresponding to the given predictor weight $\mathbf{w}$, let $\beta^d = \prod_{\ell=1}^d \|W_\ell\|$. For the kind of nets we consider i.e the ones with no bias vectors in any of the layers it follows from the definition of β that the function computed by the net remains invariant if the layer matrices W_i are replaced by $\frac{\beta}{\|W_i\|}W_i$. And we see that the spectral norm is identically β for each layer in this net with modified wights. So we can assume without loss of generality that $\forall i \, \|W_i\| = \beta$. By using this uniform norm assumption along with the assumption that

If

$$\sigma\sqrt{2h\log(4dh)} \leq \frac{\beta}{d} \tag{6.47}$$

then we have,

$$\begin{aligned}
\frac{1}{2} &\leq \mathbb{P}_{\{U_\ell \sim \mathcal{N}(0,\sigma^2 I_{h\times h})\}_{\ell=1,\ldots,d}}[\forall i \|U_\ell\| \leq \sigma\sqrt{2h\log(4dh)}] \\
&\leq \mathbb{P}\left[\|f_{\mathbf{w}+\mathbf{u}}(\mathbf{x}) - f_{\mathbf{w}}(\mathbf{x})\| \leq eB\beta^{d-1}\sum_{\ell=1}^d \|U_\ell\|\right] \qquad (6.48) \\
&\leq \mathbb{P}\left[\|f_{\mathbf{w}+\mathbf{u}}(\mathbf{x}) - f_{\mathbf{w}}(\mathbf{x})\| \leq eBd\beta^{d-1}\sigma\sqrt{2h\log(4dh)}\right] \qquad (6.49)
\end{aligned}$$

Note that the assumption (6.47) is required even in the proof by Neyshabur et al., 2017 even though it is omitted there.

We will choose the prior – used in the PAC-Bayes bound – from a finite set of distributions, $\{\pi_i = \mathcal{N}_{\mathbf{0},\sigma^2(\tilde{\beta}_i)}\}_{i=1}^K$, in a data dependent manner. Given β corresponding to the trained net $f_{\mathbf{w}}$, suppose $\exists\tilde{\beta} \in \{\tilde{\beta}_i\}_{i=1}^K$ such that $|\beta - \tilde{\beta}| \leq \frac{\beta}{d}$. $|\beta - \tilde{\beta}| \leq \frac{\beta}{d}$ also implies that $\frac{\beta^{d-1}}{e} \leq \tilde{\beta}^{d-1} \leq e\beta^{d-1}$. Furthermore, if σ satisfies the inequalities 6.50a then the condition 6.47 will hold.

$$\sigma\sqrt{2h\log(4dh)} \leq \frac{\tilde{\beta}}{de^{\frac{1}{d-1}}} \tag{6.50a}$$

$$e^2Bd\tilde{\beta}^{d-1}\sigma\sqrt{2h\log(4dh)} \leq \frac{\gamma}{4} \tag{6.50b}$$

And from equations (6.48,6.50b) we get that

$$\frac{1}{2} \leq \mathbb{P}\left[\|f_{\mathbf{w}+\mathbf{u}}(\mathbf{x}) - f_{\mathbf{w}}(\mathbf{x})\| \leq \frac{\gamma}{4}\right].$$

Therefore the condition 6.41 in Theorem 6.D.1 is satisfied. Finally we deduce from (6.50) that the largest value of σ in terms of $\tilde{\beta}$ is,

$$\sigma(\tilde{\beta}) := \min\left\{\frac{\gamma}{4e^2 Bd\tilde{\beta}^{d-1}\sqrt{2h\log(4dh)}}, \frac{\tilde{\beta}}{de^{\frac{1}{d-1}}\sqrt{2h\log(4dh)}}\right\} \tag{6.51}$$

Note that for a given neural net weight $\mathbf{w}$ (and hence the value β) the inequality event in 6.42 holds trivially in two conditions:

1. When $\beta \leq \left(\frac{\gamma}{2B}\right)^{1/d}$ this implies $\|f_{\mathbf{w}}(\mathbf{x})\| \leq \frac{\gamma}{2}$ which implies $\hat{L}_\gamma = 1$ by definition. Therefore (6.42) holds trivially.

2. From the local sensitivity analysis done above it follows that we can invoke the above theorem with $P = \mathcal{N}(0, \sigma(\tilde{\beta})^2 I)$ and $\mu_{\mathbf{w}} = \mathcal{N}(\mathbf{w}, \sigma(\tilde{\beta})^2 I)$. Which gives us the bound

$$\mathrm{KL}(\mu_{\mathbf{w}}\|P) \leq \frac{\|\mathbf{w}\|^2}{2\sigma(\tilde{\beta})^2} = \frac{\sum_{\ell=1}^d \|W_\ell\|_F^2}{2\sigma(\tilde{\beta})^2}. \tag{6.52}$$

   Note that in terms of β (which can be directly read-off from the given net $f_{\mathbf{w}}$ in Theorem 6.1.3) we have $\sigma(\tilde{\beta}) \geq \min\left\{\frac{\gamma}{4e^3 Bd\beta^{d-1}\sqrt{2h\log(4dh)}}, \frac{\beta}{de^{\frac{2}{d-1}}\sqrt{2h\log(4dh)}}\right\} = \frac{\beta\exp(-2/(d-1))}{d\sqrt{2h\log(4dh)}}\min\left\{\frac{\gamma}{4e^{3-\frac{2}{d-1}}B\beta^d}, 1\right\}$. Therefore $\mathrm{KL}(\mu_{\mathbf{w}}\|P) \leq \frac{1}{2}\frac{\sum_{\ell=1}^d \|W_\ell\|_F^2}{\beta^2}\frac{2d^2h\log(4dh)}{\exp(-\frac{4}{(d-1)})}\frac{1}{\min\left\{\frac{\gamma^2}{4^2e^{6-\frac{4}{d-1}}B^2\beta^{2d}}, 1\right\}}$. This upper bound on KL leads to the following upperbound on the square-root term in equation 6.42,

$$\sqrt{\frac{\sum_{\ell=1}^d \|W_\ell\|_F^2}{(m-1)\beta^2}\frac{d^2h\log(4dh)}{\exp(-\frac{4}{(d-1)})}\frac{1}{\min\left\{\frac{\gamma^2}{4^2e^{6-\frac{4}{d-1}}B^2\beta^{2d}}, 1\right\}} + \frac{1}{m-1}\log\frac{3m}{\delta}}$$

   We note that for any $d, h \geq 1$ we have (a) by A.M-G.M inequality $\frac{\sum_{\ell=1}^d \|W_\ell\|_F^2}{\beta^2} \geq d \geq 1$ and (b) $\exp(\frac{4}{(d-1)})d^2h\log(4dh) > 1$ (which is obvious on taking logarithm of the LHS). Therefore a sufficient condition for quantity above to be greater than 1 is that we have, $\min\left\{1, \left(\frac{\gamma}{4\cdot B\cdot\beta^d e^{3-\frac{2}{d-1}}}\right)^2\right\} \leq \frac{1}{m-1}$. And a sufficient condition for this to be true is that, $\beta \geq \left(\frac{\sqrt{m-1}\gamma}{4\exp(3-2/(d-1))B}\right)^{1/d}$.

From the above two points it follows that it suffices to prove (6.3) for,

$$\beta \in \left[\left(\frac{\gamma}{2B}\right)^{1/d}, \left(\frac{\sqrt{m-1}\gamma}{4\exp(3-2/(d-1))B}\right)^{1/d} \right]$$

.

We note that if we want a grid on the interval $[a, b]$ s.t for every value $x \in [a, b]$ there is a grid-point g s.t $|x - g| \leq \frac{x}{d}$ then a grid size of $\frac{bd}{2a}$ suffices. [6] Hence a grid of the following size K suffices for us,

$$K = \frac{d}{2} \times \left(\frac{\sqrt{m-1}}{2\exp(3-2/(d-1))}\right)^{1/d}$$

Thus the theorem we set out to prove follows by invoking Theorem 6.B.1 with the K computed above and recognizing that the set $\{\pi_i\}$ indexed by i there is our set $\{\mathcal{N}_{(0,\sigma(\tilde{\beta})^2 I)}\}$ indexed by the grid point $\tilde{\beta}$ here, Q there is our $\mu_{\mathbf{w}}$ here and the equation 6.52 above is a bound on the term $\mathrm{KL}(Q\|\pi_i)$ there. □

6.E The $\epsilon - \gamma$ lowerbound scatter plots from the experiments

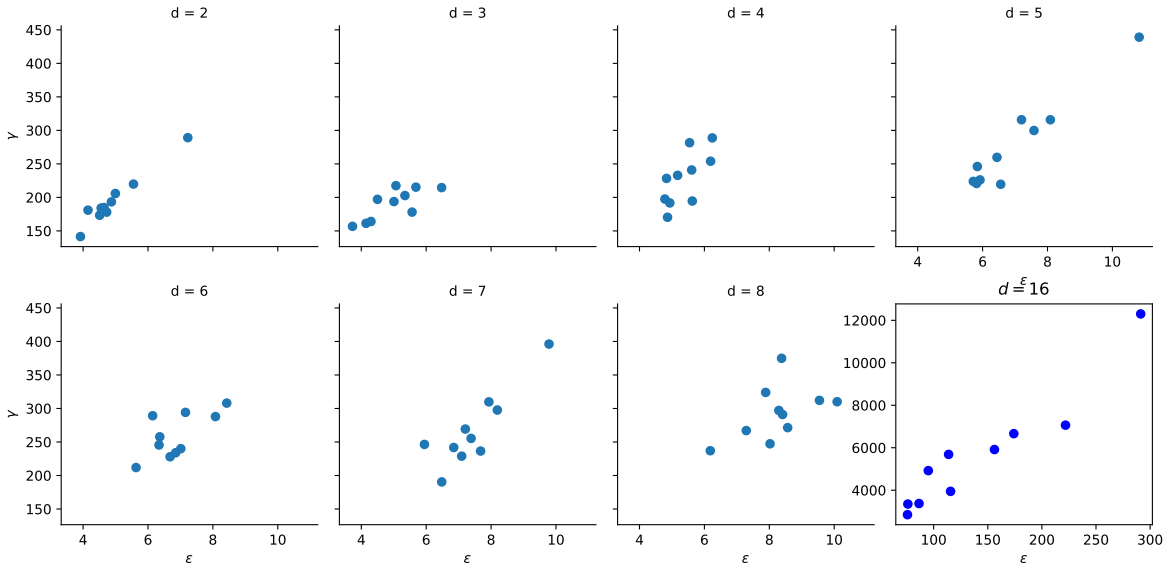


FIGURE 6.E.1: Scatter plots of the lowerbounds on ϵ and γ (as given in definition 25) while varying the depth of the net being trained on the CIFAR-10(10 trials/seeds for each)

[6] If g is the grid point which is the required approximation to x i.e $|x - g| \leq \frac{x}{d} \implies x \in \left(\frac{d}{d+1}g, \frac{d}{d-1}g\right)$ Since $a \leq g \implies \frac{2da}{d^2-1} \leq \left(\frac{d}{d-1} - \frac{d}{d+1}\right)\tilde{\beta}$. So $\frac{2da}{(d^2-1)}$ is the smallest grid spacing that might be needed and hence the maximum number number of grid points needed is $\frac{(b-a)(d^2-1)}{2ad} < \frac{(b-a)d}{2a} < \frac{bd}{2a}$

FIGURE 6.E.2: Scatter plots of the lowerbounds on ϵ and γ (as given in definition 25) for varying depth d nets trained on the the synthetic dataset for different cluster separation parameter parameter a (10 trials/seeds for each)

6.F KDE of the angular deviation during training on the synthetic dataset

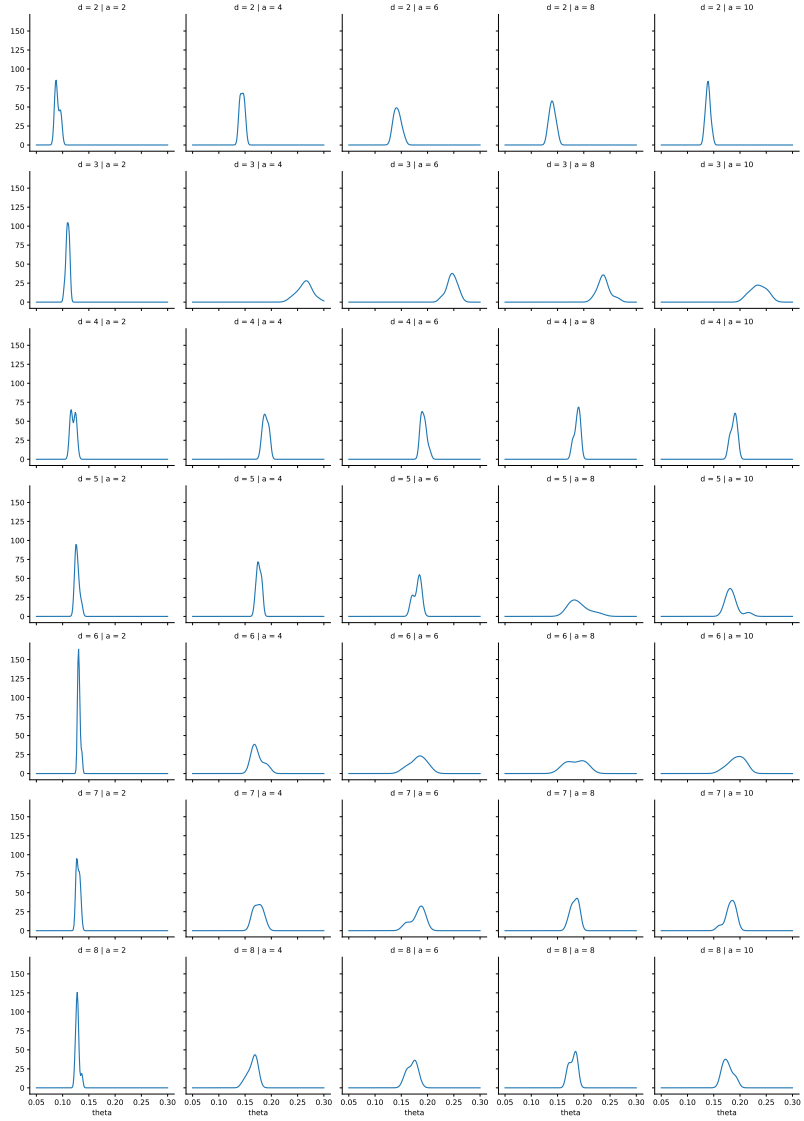


FIGURE 6.F.1: Kernel Density Estimates of the Angular Deviations θ between the initial and final networks for the differnt net depths d and cluster separation prameters a. The Y-axis shows the probability density function of the gaussian kernel density estimate.

Bibliography

Abadi, Martín et al. (2016). "TensorFlow: A System for Large-Scale Machine Learning." In: *OSDI*. Vol. 16, pp. 265–283.

Agarwal, Alekh et al. (2014). "Learning Sparsely Used Overcomplete Dictionaries." In: *COLT*, pp. 123–137.

Alain, Guillaume and Yoshua Bengio (2014). "What regularized auto-encoders learn from the data-generating distribution." In: *Journal of Machine Learning Research* 15.1, pp. 3563–3593.

Allen-Zhu, Zeyuan (2017). "Natasha 2:Faster Non-Convex Optimization Than SGD". In: *arXiv preprint arXiv:1708.08694*.

Allen-Zhu, Zeyuan and Yuanzhi Li (2019). "What Can ResNet Learn Efficiently, Going Beyond Kernels?" In: *Advances in Neural Information Processing Systems*, pp. 9015–9025.

Allen-Zhu, Zeyuan, Yuanzhi Li, and Yingyu Liang (2019). "Learning and generalization in overparameterized neural networks, going beyond two layers". In: *Advances in neural information processing systems*, pp. 6155–6166.

Allen-Zhu, Zeyuan, Yuanzhi Li, and Zhao Song (2019a). "A Convergence Theory for Deep Learning via Over-Parameterization". In: *International Conference on Machine Learning*, pp. 242–252.

— (2019b). "On the convergence rate of training recurrent neural networks". In: *Advances in Neural Information Processing Systems*, pp. 6673–6685.

Allender, Eric (1998). *Complexity Theory Lecture Notes*. https://www.cs.rutgers.edu/~allender/lecture.notes/.

Anandkumar, Animashree et al. (2014). "Tensor decompositions for learning latent variable models." In: *Journal of Machine Learning Research* 15.1, pp. 2773–2832.

Andreev, Alexander E (1987). *ABOUT ONE METHOD OF OBTAINING MORE THAN QUADRATIC EFFECTIVE LOWER BOUNDS OF COMPLEXITY OF PI-SCHEMES*.

Anthony, Martin and Peter L. Bartlett (1999). *Neural network learning: Theoretical foundations*. Cambridge University Press.

Arora, Sanjeev and Boaz Barak (2009). *Computational complexity: a modern approach*. Cambridge University Press.

Arora, Sanjeev, Rong Ge, and Ankur Moitra (2014). "New Algorithms for Learning Incoherent and Overcomplete Dictionaries." In: *COLT*, pp. 779–806.

Arora, Sanjeev et al. (2014). "More algorithms for provable dictionary learning". In: *arXiv:1401.0579*.

Arora, Sanjeev et al. (2015). "Simple, efficient, and neural algorithms for sparse coding." In: *COLT*, pp. 113–149.

Arora, Sanjeev et al. (2018). "Stronger generalization bounds for deep nets via a compression approach". In: *arXiv preprint arXiv:1802.05296*.

Arora, Sanjeev et al. (2019a). "Fine-Grained Analysis of Optimization and Generalization for Overparameterized Two-Layer Neural Networks". In: *International Conference on Machine Learning*, pp. 322–332.

Arora, Sanjeev et al. (2019b). "Harnessing the Power of Infinitely Wide Deep Nets on Small-data Tasks". In: *arXiv preprint arXiv:1910.01663*.

Arora, Sanjeev et al. (2019c). "On exact computation with an infinitely wide neural net". In: *Advances in Neural Information Processing Systems*, pp. 8139–8148.

Arpit, Devansh et al. (2015). "Why regularized auto-encoders learn sparse representation?" In: *arXiv preprint arXiv:1505.05561*.

— (2016). "Why regularized auto-encoders learn sparse representation?" In: *International Conference on Machine Learning*, pp. 136–144.

Audibert, Jean-Yves and Olivier Bousquet (2007). "Combining PAC-Bayesian and generic chaining bounds". In: *Journal of Machine Learning Research* 8.Apr, pp. 863–889.

Babanezhad, Reza et al. (2015). "Stop Wasting My Gradients: Practical SVRG". In: *arXiv preprint arXiv:1511.01942*.

Bahar, Parnia et al. (2017). "Empirical investigation of optimization algorithms in neural machine translation". In: *The Prague Bulletin of Mathematical Linguistics* 108.1, pp. 13–25.

Baldi, Pierre (2012). "Autoencoders, unsupervised learning, and deep architectures". In: *Proceedings of ICML Workshop on Unsupervised and Transfer Learning*, pp. 37–49.

Bartlett, Peter L (1998). "The sample complexity of pattern classification with neural networks: the size of the weights is more important than the size of the network". In: *IEEE transactions on Information Theory* 44.2, pp. 525–536.

Bartlett, Peter L, Dylan J Foster, and Matus J Telgarsky (2017). "Spectrally-normalized margin bounds for neural networks". In: *Advances in Neural Information Processing Systems*, pp. 6240–6249.

Bengio, Yoshua et al. (2013). "Generalized denoising auto-encoders as generative models". In: *Advances in Neural Information Processing Systems*, pp. 899–907.

Bernstein, Jeremy et al. (2018). "signSGD: compressed optimisation for non-convex problems". In: *arXiv preprint arXiv:1802.04434*.

Błasiok, Jarosław and Jelani Nelson (2016). "An improved analysis of the ER-SpUD dictionary learning algorithm". In: *arXiv:1602.05719*.

Blum, Avrim L. and Ronald L. Rivest (1992). "Training a 3-node neural network is NP-complete". In: *Neural Networks* 5.1, pp. 117–127.

Boob, Digvijay, Santanu S Dey, and Guanghui Lan (2018). "Complexity of training relu neural network". In: *arXiv preprint arXiv:1809.10787*.

Bora, Ashish et al. (2017). "Compressed Sensing using Generative Models". In: *arXiv preprint arXiv:1703.03208*.

Buhrman, Harry, Nikolay Vereshchagin, and Ronald de Wolf (2007). "On computation and communication with small bias". In: *Computational Complexity, 2007. CCC'07. Twenty-Second Annual IEEE Conference on*. IEEE, pp. 24–32.

Bun, Mark and Justin Thaler (2016). "Improved Bounds on the Sign-Rank of AC^ 0". In: *LIPIcs-Leibniz International Proceedings in Informatics*. Vol. 55. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik.

Chattopadhyay, Arkadev and Nikhil S Mande (2017). "Weights at the Bottom Matter When the Top is Heavy". In: *Electronic Colloquium on Computational Complexity, Revision 1 of Report No. 83, https://eccc.weizmann.ac.il/report/2017/083/*.

Chen, Jinghui and Quanquan Gu (2018). "Closing the generalization gap of adaptive gradient methods in training deep neural networks". In: *arXiv preprint arXiv:1806.06763*.

Chen, Ruiwen, Rahul Santhanam, and Srikanth Srinivasan (2016). "Average-case lower bounds and satisfiability algorithms for small threshold circuits". In: *LIPIcs-Leibniz International Proceedings in Informatics*. Vol. 50. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik.

Chen, Xiangyi et al. (2018). "On the convergence of a class of adam-type algorithms for non-convex optimization". In: *arXiv preprint arXiv:1808.02941*.

Chizat, Lenaic and Francis Bach (2018). "On the global convergence of gradient descent for over-parameterized models using optimal transport". In: *Advances in neural information processing systems*, pp. 3036–3046.

Coates, Adam, Andrew Ng, and Honglak Lee (2011). "An analysis of single-layer networks in unsupervised feature learning". In: *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pp. 215–223.

Coates, Adam and Andrew Y Ng (2011). "The importance of encoding versus training with sparse coding and vector quantization". In: *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pp. 921–928.

Cybenko, George (1989). "Approximation by superpositions of a sigmoidal function". In: *Mathematics of control, signals and systems* 2.4, pp. 303–314.

Dahl, George E., Tara N. Sainath, and Geoffrey E. Hinton (2013). "Improving deep neural networks for LVCSR using rectified linear units and dropout". In: *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*. IEEE, pp. 8609–8613.

Daniely, Amit (2017). "Depth Separation for Neural Networks". In: *arXiv preprint arXiv:1702.08489*.

DasGupta, Bhaskar, Hava T. Siegelmann, and Eduardo Sontag (1995). "On the complexity of training neural networks with continuous activation functions". In: *IEEE Transactions on Neural Networks* 6.6, pp. 1490–1504.

De, Soham, Anirbit Mukherjee, and Enayat Ullah (2018). "Convergence guarantees for RMSProp and ADAM in non-convex optimization and an empirical comparison to Nesterov acceleration". In: *ICML 2018 Workshop on Modern Trends in Nonconvex Optimization for Machine Learning (arXiv:1807.06766) https://tinyurl.com/y5hw79vx*.

De, Soham et al. (2017). "Automated inference with adaptive batches". In: *Artificial Intelligence and Statistics*, pp. 1504–1513.

Defazio, Aaron, Francis Bach, and Simon Lacoste-Julien (2014). "SAGA: A fast incremental gradient method with support for non-strongly convex composite objectives". In: *Advances in neural information processing systems*, pp. 1646–1654.

Denkowski, Michael and Graham Neubig (2017). "Stronger baselines for trustable results in neural machine translation". In: *arXiv preprint arXiv:1706.09733*.

Dey, Santanu S, Guanyi Wang, and Yao Xie (2018). "An Approximation Algorithm for training One-Node ReLU Neural Network". In: *arXiv preprint arXiv:1810.03592*.

Du, Simon and Jason Lee (2018). "On the Power of Over-parametrization in Neural Networks with Quadratic Activation". In: *International Conference on Machine Learning*, pp. 1329–1338.

Du, Simon S, Jason D Lee, and Yuandong Tian (2017). "When is a Convolutional Filter Easy To Learn?" In: *arXiv preprint arXiv:1709.06129*.

Du, Simon S et al. (2018). "Gradient Descent Finds Global Minima of Deep Neural Networks". In:

Duchi, John, Elad Hazan, and Yoram Singer (2011). "Adaptive subgradient methods for online learning and stochastic optimization". In: *Journal of Machine Learning Research* 12.Jul, pp. 2121–2159.

Durmus, Alain and Szymon Majewski (2019). "Analysis of Langevin Monte Carlo via Convex Optimization." In: *Journal of Machine Learning Research* 20.73, pp. 1–46.

Dziugaite, Gintare Karolina and Daniel Roy (2018a). "Entropy-SGD optimizes the prior of a PAC-Bayes bound: Generalization properties of Entropy-SGD and data-dependent priors". In: *International Conference on Machine Learning*, pp. 1376–1385.

Dziugaite, Gintare Karolina and Daniel M Roy (2017). "Computing nonvacuous generalization bounds for deep (stochastic) neural networks with many more parameters than training data". In: *arXiv preprint arXiv:1703.11008*.

— (2018b). "Data-dependent PAC-Bayes priors via differential privacy". In: *arXiv preprint arXiv:1802.09583*.

Eldan, Ronen and Ohad Shamir (2016). "The Power of Depth for Feedforward Neural Networks". In: *29th Annual Conference on Learning Theory*, pp. 907–940.

Forster, Jürgen (2002). "A linear lower bound on the unbounded error probabilistic communication complexity". In: *Journal of Computer and System Sciences* 65.4, pp. 612–625.

Forster, Jürgen et al. (2001). "Relations between communication complexity, linear arrangements, and computational complexity". In: *International Conference on Foundations of Software Technology and Theoretical Computer Science*. Springer, pp. 171–182.

Freund, Yoav and Robert E Schapire (1999). "Large margin classification using the perceptron algorithm". In: *Machine learning* 37.3, pp. 277–296.

Fridman, Lex et al. (2017). "Mit autonomous vehicle technology study: Large-scale deep learning based analysis of driver behavior and interaction with automation". In: *arXiv preprint arXiv:1711.06976*.

Gadat, Sébastien, Fabien Panloup, Sofiane Saadane, et al. (2018). "Stochastic heavy ball". In: *Electronic Journal of Statistics* 12.1, pp. 461–529.

Ge, Rong, Chi Jin, and Yi Zheng (2017). "No Spurious Local Minima in Nonconvex Low Rank Problems: A Unified Geometric Analysis". In: *arXiv preprint arXiv:1704.00708*.

Gilbert, Anna. "CBMS Conference on Sparse Approximation and Signal Recovery Algorithms, May 22-26, 2017 and 16th New Mexico Analysis Seminar, May 21". In: *https://www.math.nmsu.edu/jlakey/cbms2017/cbmslecture$_n$otes.html* ().

Gilbert, Anna C et al. (2017). "Towards Understanding the Invertibility of Convolutional Neural Networks". In: *arXiv preprint arXiv:1705.08664*.

Glorot, Xavier and Yoshua Bengio (2010). "Understanding the difficulty of training deep feedforward neural networks". In: *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pp. 249–256.

Goel, Surbhi and Adam Klivans (2017). "Learning depth-three neural networks in polynomial time". In: *arXiv preprint arXiv:1709.06010*.

Goel, Surbhi, Adam Klivans, and Raghu Meka (2018). "Learning one convolutional layer with overlapping patches". In: *arXiv preprint arXiv:1802.02547*.

Goel, Surbhi et al. (2016). "Reliably Learning the ReLU in Polynomial Time". In: *arXiv preprint arXiv:1611.10258*.

Golowich, Noah, Alexander Rakhlin, and Ohad Shamir (2018). "Size-Independent Sample Complexity of Neural Networks". In: *Conference On Learning Theory*, pp. 297–299.

Goodfellow, Ian J et al. (2013). "Maxout networks". In: *arXiv preprint arXiv:1302.4389*.

Gregor, Karol et al. (2015). "DRAW: A recurrent neural network for image generation". In: *arXiv preprint arXiv:1502.04623*.

Haeffele, Benjamin D. and René Vidal (2015). "Global optimality in tensor factorization, deep learning, and beyond". In: *arXiv preprint arXiv:1506.07540*.

Hajnal, András et al. (1987). "Threshold circuits of bounded depth". In: *Foundations of Computer Science, 1987., 28th Annual Symposium on*. IEEE, pp. 99–110.

Hanin, Boris (2017). "Universal Function Approximation by Deep Neural Nets with Bounded Width and ReLU Activations". In: *arXiv preprint arXiv:1708.02691*.

Harvey, Nick, Christopher Liaw, and Abbas Mehrabian (2017). "Nearly-tight VC-dimension bounds for piecewise linear neural networks". In: *Conference on Learning Theory*, pp. 1064–1068.

Hastad, Johan (1986). "Almost optimal lower bounds for small depth circuits". In: *Proceedings of the eighteenth annual ACM symposium on Theory of computing*. ACM, pp. 6–20.

Hinton, Geoffrey and Drew Van Camp (1993). "Keeping neural networks simple by minimizing the description length of the weights". In: *in Proc. of the 6th Ann. ACM Conf. on Computational Learning Theory*. Citeseer.

Hinton, Geoffrey et al. (2012). "Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups". In: *IEEE Signal Processing Magazine* 29.6, pp. 82–97.

Hinton, Geoffrey E., Simon Osindero, and Yee-Whye Teh (2006). "A fast learning algorithm for deep belief nets". In: *Neural computation* 18.7, pp. 1527–1554.

Hornik, Kurt (1991). "Approximation capabilities of multilayer feedforward networks". In: *Neural networks* 4.2, pp. 251–257.

Huang, Jiaoyang and Horng-Tzer Yau (2019). "Dynamics of deep neural networks and neural tangent hierarchy". In: *arXiv preprint arXiv:1909.08156*.

Impagliazzo, Russell, Raghu Meka, and David Zuckerman (2012). "Pseudorandomness from shrinkage". In: *Foundations of Computer Science (FOCS), 2012 IEEE 53rd Annual Symposium on*. IEEE, pp. 111–119.

Impagliazzo, Russell and Moni Naor (1988). "Decision trees and downward closures". In: *Structure in Complexity Theory Conference, 1988. Proceedings., Third Annual*. IEEE, pp. 29–38.

Impagliazzo, Russell, Ramamohan Paturi, and Michael E Saks (1997). "Size–Depth Tradeoffs for Threshold Circuits". In: *SIAM Journal on Computing* 26.3, pp. 693–707.

Ioffe, Sergey and Christian Szegedy (2015). "Batch normalization: Accelerating deep network training by reducing internal covariate shift". In: *arXiv preprint arXiv:1502.03167*.

Jacot, Arthur, Franck Gabriel, and Clément Hongler (2018). "Neural tangent kernel: Convergence and generalization in neural networks". In: *Advances in neural information processing systems*, pp. 8571–8580.

Janzamin, Majid, Hanie Sedghi, and Anima Anandkumar (2015). "Beating the perils of non-convexity: Guaranteed training of neural networks using tensor methods". In: *arXiv preprint arXiv:1506.08473*.

Jin, Chi, Praneeth Netrapalli, and Michael I Jordan (2017). "Accelerated Gradient Descent Escapes Saddle Points Faster than Gradient Descent". In: *arXiv preprint arXiv:1711.10456*.

Jin, Chi et al. (2018). "On the local minima of the empirical risk". In: *Advances in Neural Information Processing Systems*, pp. 4896–4905.

Johnson, Rie and Tong Zhang (2013). "Accelerating stochastic gradient descent using predictive variance reduction". In: *Advances in neural information processing systems*, pp. 315–323.

Jukna, Stasys (2012). *Boolean function complexity: advances and frontiers*. Vol. 27. Springer Science & Business Media.

Kabanets, Valentine, Daniel Kane, and Zhenjian Lu (2017). "A Polynomial Restriction Lemma with Applications." In: *Electronic Colloquium on Computational Complexity (ECCC)*. Vol. 24, p. 26.

Kakade, Sham M et al. (2011). "Efficient learning of generalized linear and single index models with isotonic regression". In: *Advances in Neural Information Processing Systems*, pp. 927–935.

Kalan, Seyed Mohammadreza Mousavi, Mahdi Soltanolkotabi, and A Salman Avestimehr (2019). "Fitting relus via sgd and quantized sgd". In: *2019 IEEE International Symposium on Information Theory (ISIT)*. IEEE, pp. 2469–2473.

Kane, Daniel M. and Ryan Williams (2015). "Super-linear gate and super-quadratic wire lower bounds for depth-two and depth-three threshold circuits". In: *arXiv preprint arXiv:1511.07860*.

Kane, Daniel M and Ryan Williams (2016). "Super-linear gate and super-quadratic wire lower bounds for depth-two and depth-three threshold circuits". In: *Proceedings of the forty-eighth annual ACM symposium on Theory of Computing*. ACM, pp. 633–643.

Kawaguchi, Kenji (2016). "Deep Learning without Poor Local Minima". In: *arXiv preprint arXiv:1605.07110*.

Kawaguchi, Kenji and Jiaoyang Huang (2019). "Gradient descent finds global minima for generalizable deep neural networks of practical sizes". In: *2019 57th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*. IEEE, pp. 92–99.

Keskar, Nitish Shirish and Richard Socher (2017). "Improving Generalization Performance by Switching from Adam to SGD". In: *arXiv preprint arXiv:1712.07628*.

Kidambi, Rahul et al. (2018). "On the insufficiency of existing momentum schemes for Stochastic Optimization". In: *International Conference on Learning Representations*. URL: https://openreview.net/forum?id=rJTutzbA-.

Kingma, Diederik P and Jimmy Ba (2014). *Adam: A Method for Stochastic Optimization. arXiv. org*.

Klivans, Adam and Raghu Meka (2017). "Learning graphical models using multiplicative weights". In: *2017 IEEE 58th Annual Symposium on Foundations of Computer Science (FOCS)*. IEEE, pp. 343–354.

Krause, Matthias and Pavel Pudlák (1994). "On the computational power of depth 2 circuits with threshold and modulo gates". In: *Proceedings of the twenty-sixth annual ACM symposium on Theory of computing*. ACM, pp. 48–57.

Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E. Hinton (2012). "Imagenet classification with deep convolutional neural networks". In: *Advances in neural information processing systems*, pp. 1097–1105.

Kuchaiev, Oleksii and Boris Ginsburg (2017). "Training Deep AutoEncoders for Collaborative Filtering". In: *arXiv preprint arXiv:1708.01715*.

Langford, John and Matthias Seeger (2001). *Bounds for averaging classifiers*. Tech. rep. Carnegie Mellon, Department of Computer Science.

Le, Quoc V. (2013). "Building high-level features using large scale unsupervised learning". In: *2013 IEEE international conference on acoustics, speech and signal processing*. IEEE, pp. 8595–8598.

LeCun, Yann, Yoshua Bengio, and Geoffrey Hinton (2015). "Deep learning". In: *Nature* 521.7553, pp. 436–444.

Ledoux, Michel and Michel Talagrand (2013). *Probability in Banach Spaces: isoperimetry and processes*. Springer Science & Business Media.

Lee, Holden, Oren Mangoubi, and Nisheeth Vishnoi (2019). "Online sampling from log-concave distributions". In: *Advances in Neural Information Processing Systems*, pp. 1226–1237.

Lee, Jaehoon et al. (2018). "Deep Neural Networks as Gaussian Processes". In:

Lee, Troy, Adi Shraibman, et al. (2009). "Lower bounds in communication complexity". In: *Foundations and Trends® in Theoretical Computer Science* 3.4, pp. 263–399.

Lehoucq, Richard B, Danny C Sorensen, and Chao Yang (1998). *ARPACK users' guide: solution of large-scale eigenvalue problems with implicitly restarted Arnoldi methods*. Vol. 6. Siam.

Li, Jian, Xuanyuan Luo, and Mingda Qiao (2019). "On generalization error bounds of noisy gradient methods for non-convex learning". In: *arXiv preprint arXiv:1902.00621*.

Li, Jun et al. (2016). "Sparseness analysis in the pretraining of deep neural networks". In: *IEEE transactions on neural networks and learning systems*.

Li, Xiaoyu and Francesco Orabona (2018). "On the Convergence of Stochastic Gradient Descent with Adaptive Stepsizes". In: *arXiv preprint arXiv:1805.08114*.

Li, Yuanzhi and Yang Yuan (2017). "Convergence Analysis of Two-layer Neural Networks with ReLU Activation". In: *arXiv preprint arXiv:1705.09886*.

Li, Zhiyuan et al. (2019). "Enhanced Convolutional Neural Tangent Kernels". In: *arXiv preprint arXiv:1911.00809*.

Liang, Shiyu and R Srikant (2016). "Why Deep Neural Networks for Function Approximation?" In:

Loizou, Nicolas and Peter Richtárik (2017). "Momentum and stochastic momentum for stochastic gradient, Newton, proximal point and subspace descent methods". In: *arXiv preprint arXiv:1712.09677*.

Lokam, Satyanarayana V et al. (2009). "Complexity lower bounds using linear algebra". In: *Foundations and Trends® in Theoretical Computer Science* 4.1–2, pp. 1–155.

Lucas, James, Richard Zemel, and Roger Grosse (2018). "Aggregated Momentum: Stability Through Passive Damping". In: *arXiv preprint arXiv:1804.00325*.

Maass, Wolfgang (1997). "Bounds for the computational power and learning complexity of analog neural nets". In: *SIAM Journal on Computing* 26.3, pp. 708–732.

Makhzani, Alireza and Brendan Frey (2013). "K-sparse autoencoders". In: *arXiv preprint arXiv:1312.5663*.

Makhzani, Alireza and Brendan J Frey (2015). "Winner-take-all autoencoders". In: *Advances in Neural Information Processing Systems*, pp. 2791–2799.

Manurangsi, Pasin and Daniel Reichman (2018). "The computational complexity of training relu (s)". In: *arXiv preprint arXiv:1810.04207*.

Martens, James and Roger Grosse (2015). "Optimizing neural networks with kronecker-factored approximate curvature". In: *International conference on machine learning*, pp. 2408–2417.

Matousek, Jiri (2002). *Lectures on discrete geometry*. Vol. 212. Springer Science & Business Media.

McAllester, David (2003). "Simplified PAC-Bayesian margin bounds". In: *Learning theory and Kernel machines*. Springer, pp. 203–215.

McAllester, David A (1999). "PAC-Bayesian model averaging". In: *Proceedings of the twelfth annual conference on Computational learning theory*. ACM, pp. 164–170.

Mei, Song, Yu Bai, and Andrea Montanari (2016). "The landscape of empirical risk for non-convex losses". In: *arXiv preprint arXiv:1607.06534*.

Melis, Gábor, Chris Dyer, and Phil Blunsom (2017). "On the state of the art of evaluation in neural language models". In: *arXiv preprint arXiv:1707.05589*.

Moitra, Ankur and Gregory Valiant (2010). "Settling the polynomial learnability of mixtures of gaussians". In: *Foundations of Computer Science (FOCS), 2010 51st Annual IEEE Symposium on*. IEEE, pp. 93–102.

Montufar, Guido F. et al. (2014). "On the number of linear regions of deep neural networks". In: *Advances in neural information processing systems*, pp. 2924–2932.

Mou, Wenlong et al. (2018). "Generalization Bounds of SGLD for Non-convex Learning: Two Theoretical Viewpoints". In: *Conference On Learning Theory*, pp. 605–638.

Mukherjee, Anirbit and Ramchandran Muthukumar (2020a). "A Study of Neural Training with Non-Gradient and Noise Assisted Gradient Methods". In: *arXiv preprint arXiv:2005.04211*.

— (2020b). "Guarantees on learning depth-2 neural networks under a data-poisoning attack". In: *arXiv preprint arXiv:2005.01699*.

Nagarajan, Vaishnavh and J Zico Kolter (2019a). "Generalization in deep networks: The role of distance from initialization". In: *arXiv preprint arXiv:1901.01672*.

Nagarajan, Vaishnavh and Zico Kolter (2019b). "Deterministic PAC-Bayesian generalization bounds for deep networks via generalizing noise-resilience". In: *International Conference on Learning Representations*. URL: https://openreview.net/forum?id=Hygn2o0qKX.

Neal, Radford M (1996). "Priors for infinite networks". In: *Bayesian Learning for Neural Networks*. Springer, pp. 29–53.

Nesterov, Yurii (1983). "A method of solving a convex programming problem with convergence rate O (1/k2)". In: *Soviet Mathematics Doklady*. Vol. 27. 2, pp. 372–376.

Neyshabur, Behnam et al. (2017). "A pac-bayesian approach to spectrally-normalized margin bounds for neural networks". In: *arXiv preprint arXiv:1707.09564*.

Ng, Andrew (2011). "Sparse autoencoder". In:

Nguyen, Thanh V, Raymond KW Wong, and Chinmay Hegde (2019). "On the dynamics of gradient descent for autoencoders". In: *The 22nd International Conference on Artificial Intelligence and Statistics*, pp. 2858–2867.

Ochs, Peter (2016). "Local Convergence of the Heavy-ball Method and iPiano for Non-convex Optimization". In: *arXiv preprint arXiv:1606.09070*.

Olshausen, Bruno A and David J Field (1996). "Emergence of simple-cell receptive field properties by learning a sparse code for natural images". In: *Nature* 381.6583, p. 607.

— (1997). "Sparse coding with an overcomplete basis set: A strategy employed by V1?" In: *Vision research* 37.23, pp. 3311–3325.

— (2005). "How close are we to understanding V1?" In: *Neural computation* 17.8, pp. 1665–1699.

O'Neill, Michael and Stephen J Wright (2017). "Behavior of accelerated gradient methods near critical points of nonconvex problems". In: *arXiv preprint arXiv:1706.07993*.

Pal, Sankar K and Sushmita Mitra (1992). "Multilayer perceptron, fuzzy sets, classifiaction". In:

Pascanu, Razvan, Guido Montufar, and Yoshua Bengio (2013). "On the number of response regions of deep feed forward networks with piece-wise linear activations". In: *arXiv preprint arXiv:1312.6098*.

Paterson, Michael S and Uri Zwick (1993). "Shrinkage of de Morgan formulae under restriction". In: *Random Structures & Algorithms* 4.2, pp. 135–150.

Polyak, Boris T (1987). "Introduction to optimization. Translations series in mathematics and engineering". In: *Optimization Software*.

Radford, Alec, Luke Metz, and Soumith Chintala (2015). "Unsupervised representation learning with deep convolutional generative adversarial networks". In: *arXiv preprint arXiv:1511.06434*.

Raghu, Maithra et al. (2016). "On the expressive power of deep neural networks". In: *arXiv preprint arXiv:1606.05336*.

Raginsky, Maxim, Alexander Rakhlin, and Matus Telgarsky (2017). "Non-convex learning via Stochastic Gradient Langevin Dynamics: a nonasymptotic analysis". In: *Conference on Learning Theory*, pp. 1674–1703.

Rangamani, Akshay et al. (2017). "Critical Points Of An Autoencoder Can Provably Recover Sparsely Used Overcomplete Dictionaries". In: *arXiv preprint arXiv:1708.03735*.

Razborov, Alexander A. (1987). "Lower bounds on the size of bounded depth circuits over a complete basis with logical addition". In: *Mathematical Notes* 41.4, pp. 333–338.

Razborov, Alexander A (1992). "On small depth threshold circuits". In: *Scandinavian Workshop on Algorithm Theory*. Springer, pp. 42–52.

Razborov, Alexander A and Alexander A Sherstov (2010). "The Sign-Rank of AC ^0". In: *SIAM Journal on Computing* 39.5, pp. 1833–1855.

Reddi, Sashank J, Satyen Kale, and Sanjiv Kumar (2018). "On the convergence of adam and beyond". In: *International Conference on Learning Representations*.

Rifai, Salah et al. (2011). "Contractive auto-encoders: Explicit invariance during feature extraction". In: *Proceedings of the 28th international conference on machine learning (ICML-11)*, pp. 833–840.

Rosenblatt, Frank (1958). "The perceptron: a probabilistic model for information storage and organization in the brain." In: *Psychological review* 65.6, p. 386.

Rossman, Benjamin (2008). "On the constant-depth complexity of k-clique". In: *Proceedings of the fortieth annual ACM symposium on Theory of computing*. ACM, pp. 721–730.

Rossman, Benjamin, Rocco A. Servedio, and Li-Yang Tan (2015). "An average-case depth hierarchy theorem for Boolean circuits". In: *Foundations of Computer Science (FOCS), 2015 IEEE 56th Annual Symposium on*. IEEE, pp. 1030–1048.

Royden, H.L. and P.M. Fitzpatrick (2010). *Real Analysis*. Prentice Hall.

Safran, Itay and Ohad Shamir (2016). "Depth separation in relu networks for approximating smooth non-linear functions". In: *arXiv preprint arXiv:1610.09887*.

— (2017). "Depth-width tradeoffs in approximating natural functions with neural networks". In: *International Conference on Machine Learning*, pp. 2979–2987.

Salakhutdinov, Ruslan and Geoffrey E. Hinton (2009). "Deep Boltzmann Machines." In: *International Conference on Artificial Intelligence and Statistics (AISTATS)*. Vol. 1, p. 3.

Saptharishi, R. (2014). *A survey of lower bounds in arithmetic circuit complexity*.

Scott, David W (2015). *Multivariate density estimation: theory, practice, and visualization*. John Wiley & Sons.

Sedghi, Hanie and Anima Anandkumar (2014). "Provable methods for training neural networks with sparse connectivity". In: *arXiv preprint arXiv:1412.2693*.

Sermanet, Pierre et al. (2014). "OverFeat: Integrated Recognition, Localization and Detection using Convolutional Networks". In: *International Conference on Learning Representations (ICLR 2014)*. arXiv preprint arXiv:1312.6229.

Serra, Thiago, Christian Tjandraatmadja, and Srikumar Ramalingam (2017). "Bounding and counting linear regions of deep neural networks". In: *arXiv preprint arXiv:1711.02114*.

Shalev-Shwartz, Shai and Shai Ben-David (2014). *Understanding machine learning: From theory to algorithms*. Cambridge university press.

Shamir, Ohad (2016). "Distribution-Specific Hardness of Learning Neural Networks". In: *arXiv preprint arXiv:1609.01037*.

Sherstov, Alexander A (2007). "Powering requires threshold depth 3". In: *Information processing letters* 102.2-3, pp. 104–107.

— (2009). "Separating AC^0 from Depth-2 Majority Circuits". In: *SIAM Journal on Computing* 38.6, pp. 2113–2129.

— (2011). "The unbounded-error communication complexity of symmetric functions". In: *Combinatorica* 31.5, pp. 583–614.

Shpilka, Amir and Amir Yehudayoff (2010). "Arithmetic circuits: A survey of recent results and open questions". In: *Foundations and Trends® in Theoretical Computer Science* 5.3–4, pp. 207–388.

Silver, David et al. (2017). "Mastering the game of go without human knowledge". In: *Nature* 550.7676, p. 354.

Silver, David et al. (2018). "A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play". In: *Science* 362.6419, pp. 1140–1144.

Simonyan, Karen and Andrew Zisserman (2014). "Very deep convolutional networks for large-scale image recognition". In: *arXiv preprint arXiv:1409.1556*.

Siu, Kai-Yeung, Vwani P Roychowdhury, and Thomas Kailath (1994). "Rational approximation techniques for analysis of neural networks". In: *IEEE Transactions on Information Theory* 40.2, pp. 455–466.

Smolensky, Roman (1987). "Algebraic methods in the theory of lower bounds for Boolean circuit complexity". In: *Proceedings of the nineteenth annual ACM symposium on Theory of computing*. ACM, pp. 77–82.

Soltanolkotabi, Mahdi (2017). "Learning relus via gradient descent". In: *Advances in neural information processing systems*, pp. 2007–2017.

Spielman, Daniel A, Huan Wang, and John Wright (2012). "Exact Recovery of Sparsely-Used Dictionaries." In: *COLT*, pp. 37–1.

Srivastava, Nitish et al. (2014). "Dropout: a simple way to prevent neural networks from overfitting." In: *Journal of Machine Learning Research* 15.1, pp. 1929–1958.

stad, Johan HÅ (1998). "The shrinkage exponent of de Morgan formulas is 2". In: *SIAM Journal on Computing* 27.1, pp. 48–64.

Staib, Matthew et al. (2019). "Escaping saddle points with adaptive gradient methods". In: *arXiv preprint arXiv:1901.09149*.

Su, Lili and Pengkun Yang (2019). "On Learning Over-parameterized Neural Networks: A Functional Approximation Perspective". In: *Advances in Neural Information Processing Systems*, pp. 2637–2646.

Subbotovskaya, Bella Abramovna (1961). "Realizations of linear functions by formulas using+". In: *Doklady Akademii Nauk SSSR* 136.3, pp. 553–555.

Sutskever, Ilya, Oriol Vinyals, and Quoc V. Le (2014). "Sequence to sequence learning with neural networks". In: *Advances in neural information processing systems*, pp. 3104–3112.

Sutskever, Ilya et al. (2013). "On the importance of initialization and momentum in deep learning". In: *International conference on machine learning*, pp. 1139–1147.

Tamaki, Suguru (2016). "A Satisfiability Algorithm for Depth Two Circuits with a Sub-Quadratic Number of Symmetric and Threshold Gates." In: *Electronic Colloquium on Computational Complexity (ECCC)*. Vol. 23. 100, p. 4.

Telgarsky, Matus (2015). "Representation Benefits of Deep Feedforward Networks". In: *arXiv preprint arXiv:1509.08101*.

— (2016a). "Benefits of depth in neural networks". In: *arXiv preprint arXiv:1602.04485*.

— (2016b). "benefits of depth in neural networks". In: *29th Annual Conference on Learning Theory*, pp. 1517–1539.

Tian, Yuandong (2017). "An Analytical Formula of Population Gradient for two-layered ReLU network and its Applications in Convergence and Critical Point Analysis". In: *arXiv preprint arXiv:1703.00560*.

Tieleman, T. and G. Hinton. "RMSprop Gradient Optimization". In: (). URL: http://www.cs.toronto.edu/\~{}tijmen/csc321/slides/lecture_slides_lec6.pdf.

Tieleman, Tijmen and Geoffrey Hinton (2012). "Lecture 6.5-RMSProp, COURSERA: Neural networks for machine learning". In: *University of Toronto, Technical Report*.

Tillmann, Andreas M (2015). "On the computational intractability of exact and approximate dictionary learning". In: *IEEE Signal Processing Letters* 22.1, pp. 45–49.

Townsend, Jamie (2008). *A new trick for calculating Jacobian vector products*. https://j-towns.github.io/2017/06/12/A-new-trick.html. [Online; accessed 17-May-2018].

Tropp, Joel A (2012). "User-friendly tail bounds for sums of random matrices". In: *Foundations of computational mathematics* 12.4, pp. 389–434.

Vardan, Papyan, Yaniv Romano, and Michael Elad (2016). "Convolutional Neural Networks Analyzed via Convolutional Sparse Coding". In: *arXiv preprint arXiv:1607.08194*.

Vaswani, Sharan, Francis Bach, and Mark Schmidt (2018). "Fast and faster convergence of SGD for over-parameterized models and an accelerated perceptron". In: *arXiv preprint arXiv:1810.07288*.

Vincent, Pascal et al. (2008). "Extracting and composing robust features with denoising autoencoders". In: *Proceedings of the 25th international conference on Machine learning*. ACM, pp. 1096–1103.

Vincent, Pascal et al. (2010). "Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion". In: *Journal of Machine Learning Research* 11.Dec, pp. 3371–3408.

Wang, Shuning (2004). "General constructive representations for continuous piecewise-linear functions". In: *IEEE Transactions on Circuits and Systems I: Regular Papers* 51.9, pp. 1889–1896.

Wang, Shuning and Xusheng Sun (2005). "Generalization of hinging hyperplanes". In: *IEEE Transactions on Information Theory* 51.12, pp. 4425–4431.

Ward, Rachel, Xiaoxia Wu, and Leon Bottou (2019). "AdaGrad stepsizes: sharp convergence over nonconvex landscapes". In: *International Conference on Machine Learning*, pp. 6677–6686.

Wei, Colin et al. (2019). "Regularization matters: Generalization and optimization of neural nets vs their induced kernel". In: *Advances in Neural Information Processing Systems*, pp. 9709–9721.

Wiegerinck, Wim, Andrzej Komoda, and Tom Heskes (1994). "Stochastic dynamics of learning with momentum in neural networks". In: *Journal of Physics A: Mathematical and General* 27.13, p. 4425.

Williams, R Ryan (2018). "Limits on representing Boolean functions by linear combinations of simple functions: thresholds, ReLUs, and low-degree polynomials". In: *arXiv preprint arXiv:1802.09121*.

Wilson, Ashia C et al. (2017). "The marginal value of adaptive gradient methods in machine learning". In: *Advances in Neural Information Processing Systems*, pp. 4151–4161.

Wu, Lei, Zhanxing Zhu, et al. (2017). "Towards Understanding Generalization of Deep Learning: Perspective of Loss Landscapes". In: *arXiv preprint arXiv:1706.10239*.

Wu, Xiaoxia, Simon S Du, and Rachel Ward (2019). "Global convergence of adaptive gradient methods for an over-parameterized neural network". In: *arXiv preprint arXiv:1902.07111*.

Xu, Pan et al. (2018). "Global convergence of langevin dynamics based algorithms for nonconvex optimization". In: *Advances in Neural Information Processing Systems*, pp. 3122–3133.

Yang, Tianbao, Qihang Lin, and Zhe Li (2016). "Unified convergence analysis of stochastic momentum methods for convex and non-convex optimization". In: *arXiv preprint arXiv:1604.03257*.

Yao, Andrew Chi-Chih (1985). "Separating the polynomial-time hierarchy by oracles". In: *Foundations of Computer Science, 1985., 26th Annual Symposium on*. IEEE, pp. 1–10.

Yarotsky, Dmitry (2016). "Error bounds for approximations with deep ReLU networks". In: *arXiv preprint arXiv:1610.01145*.

Yuan, Kun, Bicheng Ying, and Ali H Sayed (2016). "On the influence of momentum acceleration on online learning". In: *Journal of Machine Learning Research* 17.192, pp. 1–66.

Zaheer, Manzil et al. (2018). "Adaptive Methods for Nonconvex Optimization". In: *Advances in Neural Information Processing Systems*.

Zavriev, SK and FV Kostyuk (1993). "Heavy-ball method in nonconvex optimization problems". In: *Computational Mathematics and Modeling* 4.4, pp. 336–341.

Zhang, Qiuyi et al. (2017). "Electron-Proton Dynamics in Deep Learning". In: *arXiv preprint arXiv:1702.00458*.

Zhang, Yuchen, Percy Liang, and Moses Charikar (2017). "A Hitting Time Analysis of Stochastic Gradient Langevin Dynamics". In: *Proceedings of Machine Learning Research vol* 65, pp. 1–43.

Zhou, Dongruo et al. (2018a). "On the convergence of adaptive gradient methods for nonconvex optimization". In: *arXiv preprint arXiv:1808.05671*.

Zhou, Wenda et al. (2018b). "Non-vacuous generalization bounds at the imagenet scale: a PAC-bayesian compression approach". In:

Ziegler, Günter M. (1995). *Lectures on polytopes*. Vol. 152. Springer Science & Business Media.

Zou, Difan and Quanquan Gu (2019). "An improved analysis of training over-parameterized deep neural networks". In: *Advances in Neural Information Processing Systems*, pp. 2053–2062.

Zou, Difan et al. (2018a). "Stochastic gradient descent optimizes over-parameterized deep relu networks". In: *arXiv preprint arXiv:1811.08888*.

Zou, Fangyu et al. (2018b). "A Sufficient Condition for Convergences of Adam and RMSProp". In: *arXiv preprint arXiv:1811.09358*.

Curriculum Vitae

Education

Fall 2020-	Post-Doctoral Researcher at UPenn (Wharton), Statistics with Prof. Weijie Su
Spring 2016	Ph.D. student (Research Assistant) at the Department of Applied Mathematics
-Spring 2020	and Statistics, Johns Hopkins University (JHU),
	`Adviser :` Prof. Amitabh Basu
Fall 2011	Graduate student (Physics) University of Illinois at Urbana-Champaign
-Spring 2015	(UIUC)`(Starting Fall 2014 I was a TA in the CS, UIUC department)`
2008-2011	Masters of Science (Physics) Tata Institute of Fundamental Research (TIFR)
2005-2008	Bachelor of Science (Hons.) Chennai Mathematical Institute (CMI)

Internships

June-August 2019	**Research Intern Adobe Research in San Jose** Worked with Prof. Sridhar Mahadevan and Anup Rao on distribution free training of deep-nets
July-August 2018	**Research Intern Vector Institute For Artificial Intelligence, UToronto** Work with Prof. Dan Roy on PAC-Bayesian and compression based approaches to proving neural risk bounds

Research Awards

ITA 2020	**"ITA Sea Award" in recognition of the talk delivered at the conference in the "Graduation Day" track.**
JHU	**Walter L. Robb Fellowship for 2018-2019**
JHU	**Mathematical Institute for Data Science (MINDS) Fellowship for 2019-2020**

Professional Service

Journal reviewer	IEEE Transactions on Pattern Analysis and Machine Intelligence, IEEE Signal Processing Magazine, Neural Computation
Reviewer	COLT 2019, AISTATS 2018-2019, ICML 2020 NeurIPS 2018-19 ("top 50%" reviewer recognition), ICLR 2019
Program committee	AAAI-2020, ICML 2018 : Modern Trends in Nonconvex Optimization for Machine Learning, NeurIPS 2019 : Machine Learning with Guarantees
Session chair	INFORMS 2020 : Session on Deep Learning Theory and Causal Inference
Societies	Institute sponsored student member for $2018-2020$ to the "Association for Women in Mathematics"

Completed Papers

(link) Google Scholar Profile

(link) (arXiv:2005.04211) "A Study of Neural Training with Non-Gradient and Noise Assisted Gradient Methods" with Ramchandran Muthukumar

(link) (arXiv:2005.01699) "Guarantees on Learning Depth-2 Neural Networks Under a Data Poisoning Attack" with Ramchandran Muthukumar

(link) "Improving PAC-Bayesian risk bounds on deep nets using geometrical properties of the training algorithm"ICML 2019 Workshop, Understanding and Improving Generalization in Deep Learningwith Pushpendre Rastogi, Dan Roy (Vector, UToronto) and Jun Yang (Vector, UToronto)

(link) (arXiv:1807.06766)"Convergence guarantees for RMSProp and ADAM in non-convex optimization and an empirical comparison to Nesterov acceleration"ICML 2018 Workshop, Modern Trends in Nonconvex Optimization for Machine Learningwith Soham De (UMD), Enayat Ullah

(link) (ECCC/TR-17/190) "Lower bounds over Boolean inputs for deep neural networks with ReLU gates"with Amitabh Basu

(link) (arXiv:1708.03735) "Sparse coding and Autoencoders" IEEE International Symposium on Information Theory (ISIT), 2018 and NIPS 2017 workshop on "Deep Learning : Bridging Theory and Practice" carry variations of this [link] shorter version of this otherwise long calculationwith Akshay Rangamani, Amitabh Basu, Tejaswini Ganapathy (Salesforce, San Francisco), Ashish Arora, Trac D.Tran, Sang (Peter) Chin (BU)

(link) (ECCC/TR-17/098) "Understanding Deep Neural Networks with Rectified Linear Units" International Conference on Learning Representations (ICLR), 2018with Raman Arora, Amitabh Basu and Poorya Mianjy

(link) (arXiv:1512.01226) "Renyi entropy of the critical O(N) model"

(link) (arXiv:1307.7714) "n-point correlations of dark matter tracers : Renormalization with univariate biasing and its $O(f_{NL})$ terms with bivariate biasing"

Major exploratory projects undertaken before coming to JHU

UIUC `Ramanujan expanders`, We developed deterministic polynomial time heuristics to construct optimal spectral expanders.[Link to the research notes and expository surveys that we have written]

TIFR `Supersymmetry`, I contributed to a paper on the single particle spectrum of N = 3 superconformal Chern-Simons's theories with adjoint and fundamental matter in the large N_c limit.

Invited conference talks

6^{th} May 2020 (postponed) SIAM Conference on Mathematics of Data Science 2020

5^{th} Feb 2020 **"Graduation Day" talk at the 2020 Information Theory and Applications Workshop**

20^{th} Oct 2019 **(2 talks) INFORMS Annual Meeting 2019, Seattle**

4^{th} Nov 2018 **INFORMS Annual Meeting 2018, Phoenix**

9^{th} Jul. 2018 **SIAM Annual Meeting 2018, Portland**

1^{st} Jul 2018 23^{rd} **International Symposium on Mathematical Programming (ISMP) (Bordeux, France)**

17^{th} Aug 2017 **"Modeling and Optimization: Theory and Applications" (MOPTA) 2017**

Talks delivered at institutes during visits

21^{st} May 2020 **University of Utah, School of Computing**Provable training of depth-2 nets.

21^{st} Jan 2020 **NVIDIA AI Lab, Toronto** Mathematics of training small neural nets and their PAC-Bayesian risk bounds.

22^{th} Nov 2019 **University of Michigan, Biostatistics** Mathematics of training small neural nets and their PAC-Bayesian risk bounds.

13^{th} Nov 2019 **Harvard University, Applied Mathematics** Mathematics of training small neural nets and their PAC-Bayesian risk bounds.

22^{th} Aug 2019 **Google Brain, Mountain View** Mathematics of training small neural nets and their PAC-Bayesian risk bounds. **[Slides at this link]**

16^{th} Jan 2019 **The Institute of Mathematical Sciences** A 2 hour long deep-learning survey talk as a part of their graduate course "Mathematics Of ML". **[Slides at this link]**

25^{th} July 2018 **Vector Institute For Artificial Intelligence** A 2 hour long survey talk on our theorems on deep learning. **[Slides at this link]**

14^{th} Nov 2017 **Massachusetts Institute of Technology (MIT), Department of Mathematics** A 1 hour survey talk on our deep net theorems.(contents absorbed into the above survey)

1^{st} Dec 2016 **Indian Institute of Science Education and Research (IISER), Pune** Understanding deep neural networks with ReLU activation.

28^{th} Sep 2015 **IBM Research Laboratory, Delhi (Bangalore)** Can we construct Ramanujan graphs?**[Slides at this link]**

2^{nd} Sep 2015 **Johns Hopkins University** Possible approaches towards constructing Ramanujan graphs in deterministic polynomial time.

2^{nd} Apr 2015 **UIUC** Sum of squares proof system and hypercontractive inequality for characterizing small-set expansion.**[Slides at this link]**

Jun 2014 **Indian Institute of Science (IISc.)** Progress on understanding Klebanov's conjecture about Renyi entropy of the criticial $O(N)$-model.

Funding awarded to attend conferences

29^{th} Oct -2^{nd} Nov 2018 Simons' Institute (Berkeley) workshop on "Robust and High-Dimensional Statistics", Duncan Fund

17^{th}-22^{nd} June 2018 IEEE International Symposium on Information Theory (ISIT), 2018
Student Travel Award

30^{th} April-3^{rd} May 2018 International Conference on Learning Representations (ICLR), 2018
Student Travel Award

22^{nd}-23^{rd} June 2016 Chaining Methods and their Applications to Computer Science, Harvard University I received funding from NSF (National Science Foundation) to attend this workshop.

24^{th} June 2016 Theory of Deep Learning, ICML 2016
I received departmental funding to attend this conference.

12^{th}-15^{th} June 2010 I was selected to attended the International Congress of Mathematicians (ICM) satellite meeting on "Geometric Topology and Riemannian Geometry" at the Indian Institute of Science, Bangalore, India

Major graduate courses completed in mathematics, computer science and physics

Mathematics	**Algebraic Geometry :** Algebraic curves and Riemann surfaces
	Optimization : Introduction to Convexity, Sparse Recovery and Compressed Sensing, Nonlinear Optimization II, Machine Learning
	Probability : Probability Theory I, High dimensional probability and statistical learning theory, Stochastic Calculus
	Analysis : Introduction to Harmonic Analysis and Its Applications
Computer Science	**Theoretical CS :** Algorithms, Computational Complexity, Spectral Graph Theory, Advanced Cryptography, Pseudorandomness
Physics	**Field Theory :** Quantum Field Theory , Particle physics, Phase transitions and Critical Phenomenon

Undergraduate awards

2008	**Lindau conference,** Selected as a member of the Indian contingent to The Nobel Laureate Meetings at Lindau (Physics). I delivered a presentation at the DFG Headquarters in Bonn, Germany on behalf of the Indian contingent
2003-2008	**KVPY,** I was among the 40 students across India who were awarded the Kishore Vaigyanik Protsahan Yojana (KVPY) scholarship, from the Department of Science and Technology, Govt. of India. In India at the undergraduate level KVPY scholarship has the highest monetary value.
2006	**CalTech,** Selected to the California Institute of Technology as an international transfer student. Selection was through 2 subjective tests in Physics and Mathematics
2006	**SINP,** Selected to the "Undergraduate Associateship" in Physics at the Saha Institute of Theoretical Physics (SINP) which provided me an annual travel and book grant for 2 years

High-school achievements

2005 **ISI,** Selected to the prestigious B.Stat program of the Indian Statistical Institute (ISI) through one of the most competitive nationwide mathematics examination consisting of two levels of written testing and an hour long problem solving interview

2005 **Presidency College, Kolkata,** Secured rank 1 and 4 at the undergraduate physics and mathematics entrance tests of the Presidency College, Kolkata.

2002 **NSO,** Secured All India Rank 35 in the 4^{th} National Science Olympiad (NSO)